UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INFORMÁTICA SUPERIOR

PROYECTO FIN DE CARRERA

FUNCIONALIDADES DE AUDIO DE KINECT

VÍDEO ACTA: UNA PRUEBA DE CONCEPTO

Proyecto Fin de Carrera presentado por:

Ricardo Vílchez García

Dirigido por:

D. Miguel Ángel Patricio Guisado
D. Álvaro Luis Bustamante

Julio, 2012

Valar Morghulis

Canción de Hielo y Fuego, George RR Martin

The more I know people, the more I love my dog

Mark Twain

Agradecimientos

Quisiera mostrar mi agradecimiento a la Universidad Carlos III por la excelente formación que he recibido durante los años que he cursado la carrera.

También a mi familia, que siempre me ha apoyado en todo lo que me he propuesto.

Contenido

Índice de ilustraciones

1. Capítulo 1: Introducción

En este primer capítulo se va a realizar una presentación al contexto en el que se desarrolla el documento, los objetivos que lo guían, se describe un pequeño glosario de términos y se muestra la estructura general del proyecto.

1.1. Un poco de historia

Cuando inicié mis estudios en Ingeniería Informática Superior no tenía una idea clara de en qué me estaba metiendo, sinceramente. Durante los primeros cursos el temario era muy genérico en el ámbito de la Informática, sobre todo el primer curso. Asignaturas como Cálculo, Álgebra, Lógica Computacional, Matemática Discreta o Física son, con diferencias, comunes en el primer año la mayoría de las Ingenierías.

El segundo curso fue de todos el más intenso. El cambio desde primer curso fue, en mi opinión, mucho más drástico que de segundo de Bachillerato a primero de carrera. Drástico en el sentido de que la carrera entraba realmente en materia: asignaturas como Estructuras de Datos, Sistemas Operativos, Programación o Ficheros introdujeron temario puro de Informática y fue el auténtico punto de contacto con el sector. Fue además, dada la carga de trabajo de las asignaturas, el año más sufrido de todos.

Segundo dio paso a un tercero mucho más interesante. Tercero es el año en el que el estudiante debe escoger especialidad. En mi caso lo tenía claro, mi meta era trabajar en "algo" relacionado con la Inteligencia Artificial. No tenía nada claro qué podía ser ese "algo", pero sí que sabía que no estaba en las otras dos especializaciones: Desarrollo de sistemas de información en la empresa y Aplicaciones y sistemas distribuidos. Noté que asignaturas como Interfaces de usuario o Seguridad en tecnologías de la información me llamaban más la atención que el resto y empecé a definir en mi cabeza la forma de ese "algo".

Cuarto fue un año bastante interesante a nivel académico, sobre todo por asignaturas como Inteligencia Artificial o Robótica. Fue durante este año cuando decidí, motivado por relatos de conocidos y compañeros, a realizar un intercambio académico en otro país. Me apunté por tanto al programa Erasmus y tuve la suerte de obtener una plaza para estudiar un curso completo en la Universidad Politécnica de Vilnius, Lituania. Lo que debería haber sido mi último año (quinto) se convirtió en la mejor experiencia de mi vida a nivel personal.

Antes de volver de Vilnius decidí que no quería acabar mi último año en Madrid, por lo que me inscribí a otro programa de intercambio, en este caso a nivel nacional, la beca SICUE, más conocida por la vertiente económica como beca Séneca.

Afortunado de nuevo, me concedieron una plaza de un cuatrimestre (no disponía de más créditos) en la Universidad Politécnica de Cataluña, donde cursé asignaturas troncales y créditos de libre elección que me quedaban. Fue aquí donde cursé la asignatura de Videojuegos, para mí la más cercana junto con Inteligencia Artificial a mis preferencias profesionales (ese "algo" fantástico).

Al volver en el segundo cuatrimestre a Colmenarejo contacté con mi antiguo profesor Miguel Ángel Patricio y le consulté sobre los proyectos fin de carrera que tenía pensados. Elegí a Miguel Ángel por dos razones: ya le conocía de antes al haber sido alumno suyo en un par de asignaturas que me gustaron, y sabía era miembro del Grupo de Inteligencia Artificial Aplicada (GIAA) de la universidad. Este laboratorio lleva proyectos informáticos relacionados con el área de Inteligencia Artificial para importantes empresas tecnológicas y fue mi primera opción a la hora de buscar PFC.

De los proyectos fin de carrera que Miguel Ángel Patricio me propuso el que más me llamó la atención era uno que tenía pendiente y que implicaba el uso del sensor Kinect de Microsoft, más conocido como "la cámara de Xbox 360", la popular videoconsola. Lo que yo desconocía, o al menos tenía poca constancia, era que Kinect llevaba algo más de un año ganando importancia en el sector de la investigación científica, sobre todo haciendo uso de su sensor de profundidad como alternativa a las caras cámaras de tiempo de vuelo.

Como buen representante de mi generación, y aspirante a ingeniero informático para más inri, me llamó la atención. La idea era estudiar las posibilidades del sensor en cuanto a las capacidades de recepción y tratamiento de audio, una faceta del sensor relegada a segunda posición debido a su cámara de profundidad.

El momento coincidía (apenas una semana después) con el lanzamiento de la primera versión del SDK de Kinect para Windows. Esto significaba que Microsoft acababa de lanzar al público la primera versión comercial del conjunto de librerías necesarias para programar aplicaciones que usaran las funcionalidades de Kinect, una herramienta de valor aún no calculado y que esperaba ser explotada, sobre todo por la comunidad de investigación tecnológica.

Acepté el PFC y este documento es el resultado. A lo largo de sus páginas se describe el trabajo realizado de investigación, estudio y prueba del sensor Kinect.

1.2. Ambiente

Vivimos en un momento impresionante. Hace diez años las novedades se anunciaban cada año, hoy en día es raro el día en que no se anuncie una novedad: La tecnología nos fascina cada día. Cosas que hace 10 años eran ciencia ficción, hace 5 pasaron a ser ideas y en la actualidad son realidad.

Un ejemplo son las interfaces de usuario, componentes que proporcionan interacción entre usuarios y ordenadores. Desde sus primitivas formas en los años 70 como *Interfaces por Línea de Comandos*, evolucionando a las *Interfaces Gráficas de Usuario* durante los años 80 y 90, el último paso en su evolución es el que nos ocupa en este documento, las *Interfaces Naturales de Usuario.*

Con las Interfaces Naturales de Usuario se pretende desprenderse del concepto de interfaces y dispositivos cableados para interactuar con ordenadores y empezar a interactuar de una forma natural, usando para ello movimientos del cuerpo y usando nuestra voz.

Bajo esta filosofía llegaba al público por parte de la compañía Microsoft el sensor Kinect a finales del año 2010 para la videoconsola Xbox 360, un dispositivo innovador y que seguía a precedentes como el sensor Wiimote para la videoconsola Wii o el sensor Playstation Move para Playstation 3.

A diferencia de estos, que en sí consistían en un mando que el jugador debía usar manualmente, Kinect se coloca bajo el televisor, en frente del jugador. Las acciones de movimiento y determinados comandos de voz son lo único necesario para interactuar con el sensor.

Esta fue la teoría. Sin embargo, lo más fascinante (y al mismo tiempo predecible) es que la gente rápidamente vio el potencial que este dispositivo tenía y comenzó, despacio primero y con más recursos en adelante, a realizar proyectos usando Kinect que nada tenían que ver con las aplicaciones comerciales para videojuegos con las que Microsoft lo lanzó al mercado.

Fue el espíritu de innovación el que provocó que Kinect sea en la actualidad uno de los productos comerciales más usados para investigación en proyectos tecnológicos que requieran interacción con usuarios. Este ambiente dinámico define el contexto de este documento.

Otra cuestión hoy en día es la seguridad electrónica. ¿Cómo se asegura que un documento, un fichero de audio, video, una presentación, cualquier fichero electrónico en general no ha sido alterado desde que fue creado? ¿Cómo asegurar que la persona que dice ser su autor lo es realmente?

Estas preguntas describen los conceptos de integridad y autenticidad sobre ficheros electrónicos. Sobre estos conceptos se tratará en este documento.

1.3. Objetivos

Este Proyecto Fin de Carrera tiene como objetivos principales:

- Realizar una descripción de las funcionalidades integradas en el dispositivo Kinect para la obtención de información de una fuente de sonido dentro de su rango de funcionamiento.
- Realizar un análisis de la sensibilidad del sensor Kinect a la hora de calcular la localización de fuente sonora: el ángulo respecto al sensor Kinect en el que se sitúa la persona hablando.
- Desarrollo de una prueba de concepto en la que se demuestren estas funcionalidades de audio en una aplicación práctica. Esta aplicación consistirá en la obtención de un video acta y se deberá proveer de mecanismos para garantizar su autenticidad e integridad.

1.4. Terminología

En este apartado se describen algunos términos usados en el documento.

- **Entorno de ejecución** *(runtime environment*) es un estado de máquina virtual que suministra servicios de software para procesos o programas mientras que una computadora se está ejecutando. Puede pertenecer al mismo sistema operativo, o al software que funciona debajo de ella. Para el ámbito de este documento, que es un entorno Microsoft Windows y el marco de trabajo .NET, el entorno de ejecución que se usará es Common Language Runtime (CLR).
- **SDK**: Un SDK (siglas de *Software Development Kit* en inglés), kit de desarrollo de software en español) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, como pueden ser ciertos paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, u otras plataformas. Puede ser algo tan sencillo como una interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*) creada para permitir el uso de cierto lenguaje de programación, o puede, por ejemplo, incluir hardware sofisticado para comunicarse con un determinado sistema embebido.
- **FPS**: Frames per Second: Es una medida usada comúnmente para medir la velocidad de un sistema de representar imágenes por segundo.

1.5. Estructura del documento

Este documento tiene una estructurada dividida en apartados:

1. **Capítulo 1: Introducción**: Sirve de introducción al resto del documento. Presenta el contexto sobre el que se desarrolla el mismo, sus objetivos principales y la terminología usada. El resto de apartados dedican un apartado a cada objetivo principal del proyecto.

2. **Capítulo 2: Sensor Kinect.** Dedicado íntegramente al sensor Kinect. En él se describe de forma general el mismo y de proporciona un estado del arte sobre el estado actual del desarrollo de proyectos que hacen uso de él. Se proporcionarán ejemplos de aplicaciones que usan el sensor de profundidad del sensor.

3. **Capítulo 3: Kinect for Windows SDK:** Centra la investigación principal que se ha realizado en el presente PFC. En él se define el concepto sobre el que gira Kinect, las interfaces de usuario naturales, donde se explica la evolución que ha tenido a lo largo de las últimas décadas. Posteriormente se definen las características de Kinect que son accesibles a los desarrolladores a través del SDK, del que se hace un pequeño análisis de las versiones que han sido lanzadas hasta las fecha.

4. **Capítulo 4: Análisis de sensibilidad:** La funcionalidad de localización de una fuente sonora, la más importante de Kinect desde el punto de vista de audio, es la evaluada en este apartado. Se describirán las pruebas realizadas para evaluar la precisión y la calidad de la cámara la hora de realizar este proceso.

5. **Capítulo 5: Prueba de concepto.** Kinect es un dispositivo asombroso, y no sólo por sus capacidades relacionadas con la imagen. A modo de demostración de lo que es posible hacer con él tratando audio, el quinto apartado describe una aplicación de escritorio que se ha desarrollado a modo de aplicación de referencia para que investigadores, curiosos, interesados o apasionados de las nuevas tecnologías vean lo fácil y divertido que resulta programar para Kinect y el gran potencial que tiene el uso de este sensor haciendo hincapié en las funciones de audio.

6. **Capítulo 6: Conclusiones.** Finalmente se incluye un apartado de conclusiones en el que se da fe del trabajo realizado y se proponen propuestas de futuro sobre este PFC así como posibles mejoras. Además se incluyen las referencias más importantes que se han usado y seguido para la realización de este documento.

7. **Capítulo 7: Gestión del proyecto**: Se recogen cuestiones como la planificación realizada o la estimación del coste del proyecto.

2. Capítulo 2: Sensor Kinect

En este capítulo nos vamos a centrar en describir el dispositivo sobre el que centra este PFC: Kinect. En el apartado *2.1.Introducción* se explica qué hace especial a Kinect y cómo Microsoft decidió proporcionar un SDK para desarrollar aplicaciones sobre Kinect en Windows, algo que Microsoft no tenía planeado hacer cuando lanzó lo que fue pensado como un nuevo tipo de "control libre" para el sistema Xbox.

2.1. Introducción

Kinect es un periférico de entrada que actúa como sensor de movimiento creado por la compañía Microsoft para la videoconsola Xbox 360 y para PC con sistema operativo Windows. Basado en una cámara web y pensado como un periférico para ambas plataformas, permite a los usuarios, por ejemplo, controlar e interactuar con la consola Xbox 360 sin necesidad de tocar un dispositivo de juego, a través de una interfaz de usuario natural usando gestos y comandos de voz.

Ilustración 1 Sensor Kinect para Xbox 360

El proyecto de Kinect estaba dirigido en un principio a ampliar la audiencia de la Xbox 360 más allá de su base de jugador típico:

> *"Microsoft wants people to start using their full bodies to play video games. The Redmond software giant showed off a sensor-based technology that recognizes faces, voices and body joints to affect the movements on screen. In a demo game called Ricochet, players can use their arms, legs, torso and head to block an onslaught of virtual projectiles. Another game called Paint Party lets users splash virtual paint onto an on-screen canvas. Players call out different colors to change the palette.*
>
> *The effort aims to attract a broader audience to Microsoft's console. Most of the 30 million Xbox 360s sold since November 2005 has been snapped up by avid young males drawn to complex shooter or adventure games such as Halo and Gears of War".*

Kinect para Xbox 360 fue la respuesta de Microsoft al popular Wiimote de Nintendo para Wii y al sistema de control PlayStation Move de Sony para Playstation 3, llevando los videojuegos a un nivel de interacción nunca antes visto, al prescindir por completo del mando de control.

Ilustración 2 Sensor Wiimote para Wii

Ilustración 3 Sensor Playstation Move para Playstation 3

Microsoft anunció una versión para sistemas operativos Windows el 1 de febrero de 2012, al mismo tiempo que se anunció la nueva versión de *Kinect for Windows SDK 1.0*.

2.2. La creación de Kinect

La historia de Kinect comienza mucho antes de que el propio dispositivo fuera concebido. Kinect es el resultado de décadas de análisis e investigación sobre interfaces de usuario basadas en gestos y voz. El gran éxito de la película *Minority Report* en el año 2002 añadió más leña al fuego con su representación futurista de una interfaz de usuario natural sin controladores. A partir de ahí, la rivalidad entre las consolas de videojuegos para diseñar controladoras naturales de calidad ha traído la tecnología Kinect a los salones del gran público. Fue sin embargo la filosofía *hacker* de desbloquear todo aquello con intención de ser ocultado o escondido la que eventualmente abrió las puertas de Kinect a los desarrolladores.

Precedentes

Bill Buxton, uno de los padres de la interacción hombre-computador, ha estado hablando durante los últimos años acerca de lo que él llama *The Long Nose of Innovation*, en referencia al término *The Long Tail* acuñado por Chris Anderson, director de la revista *Wired. The Long Nose* describe las décadas de incubación necesarias para producir

una nueva tecnología "revolucionaria" aparentemente de la nada. El clásico ejemplo de este concepto es la invención y el perfeccionamiento del dispositivo principal de la revolución que supuso la interfaz gráfica de usuario: el ratón.

El primer ratón fue construido por Douglas Engelbart y Bill English en el Stanford Research Institute en 1963. Incluso le dieron su nombre al dispositivo de ratón. Bill English llevó el concepto desarrollado aún más allá cuando se la llevó a Xerox PARC en 1973. Con Jack Hawley, agregó la famosa bola del ratón al diseño del ratón. Durante este mismo período de tiempo, Telefunken en Alemania desarrolló independientemente su propio dispositivo de ratón de bola rodante llamado Telefunken Rollkugel. En 1982, el primer ratón comercial comenzó a abrirse camino en el mercado. Logitech comenzó a venderlos por 299 dólares americanos. Fue en este período cuando Steve Jobs visitó Xerox PARC y vio el ratón trabajando con una interfaz WIMP (ventanas, iconos, menús, punteros). Algún tiempo después de eso, Jobs invitó a Bill Gates para enseñarle en interfaz gráfica de usuario basada en el ratón en la que estaba trabajando. Apple lanzó el computador Lisa en 1983 con un ratón, y posteriormente el Macintosh incorporó el ratón en el año 1984. Microsoft anunció su sistema operativo Windows poco después de Lisa y comenzó a vender Windows 1.0 en 1985. No fue sino hasta 1995, con el lanzamiento de la versión del sistema operativo Windows 95, cuando el ratón se convirtió en omnipresente. *The Long Nose* describe los 30 años necesarios para que dispositivos como el ratón pasen de ser una invención a ser omnipresentes en la industria.

Un estudio similar de 30 años que describe *The Long Nose* puede ser aplicado para Kinect. A partir de finales de los años 70, a mitad de camino entre la trayectoria de desarrollo del ratón, Chris Schmandt en el Architectural Machine Group del MIT comenzó un proyecto de investigación llamado *Put-That-Here*, basado en una idea de Richard Bolt de reconocimiento combinado de voz y gesto como vectores de entrada de un interfaz gráfica. El proyecto *Put-That-Here* estaba instalado en una habitación de dieciséis por once pies con una gran pantalla de proyección contra una pared. El usuario se sentaba en una silla de vinilo a unos ocho metros de frente a la pantalla y tenía un pequeño dispositivo magnético escondido en una muñeca a modo de entrada de movimiento espacial, así como un micrófono montado en la cabeza. Con estas entradas, y un poco de lógica elemental en torno a reconocimiento de pronombres como "eso" o "aquí", el usuario podía crear y mover formas básicas sobre la pantalla. Bolt sugiere en 1980 en su artículo que describe el proyecto, "*Put-That-Here*: *Voice and Gesture at the Graphics Interface*", que con el tiempo el micrófono montado en la cabeza del usuario debe ser remplazado con un micrófono direccional. Versiones posteriores de *Put-That-Here* permitían a los usuarios guiar a los barcos a través del mar Caribe o colocar edificios coloniales en el mapa de Boston.

Otro proyecto de investigación del MIT Media Labs en el año 1993 formado por David Koonz, Thorrison Kristinn, y Carlton Sparrell (y de nuevo dirigido por Bolt) llamado *The Iconic System* refinó el concepto de *Put-That-Here* para poder trabajar con palabras y gestos, así como una tercera modalidad de entrada: movimiento del ojo. Además, en lugar de proyectar la entrada en un espacio bidimensional, la interfaz gráfica era un espacio en tres dimensiones generada por ordenador. En lugar de los cubos magnéticos utilizados con *Put-That-Here*, *The Iconic System* utilizaba un sistema de guantes especiales para facilitar el seguimiento de los gestos de usuario.

Hacia finales de los 90, Marcos Lucente desarrolló una interfaz de usuario avanzada para IBM Research llamada *DreamSpace*, que funcionó en variedad de plataformas incluyendo Windows NT. Incluso implementaba la sintaxis de *Put-That-Here* de 1979 de Chris Schmandt en el proyecto. A diferencia de cualquiera de sus predecesores, *Dreamspace* no utilizaba tubos o guantes para el reconocimiento de gestos. En su lugar, utilizaba un sistema de visión. Además, Lucente pensó en *Dreamspace* no sólo para escenarios especializados, sino también como una alternativa viable al ratón estándar y entradas de teclado para la informática cotidiana. Lucente ayudó popularizar el reconocimiento de gestos y de voz mediante la demostración de *Dreamspace* en ferias entre 1997 a 1999.

En 1999, John Underkoffler (también del MIT Media Labs y coautor junto con Mark Lucente unos pocos años antes de un estudio sobre holografía) fue invitado a trabajar en un nuevo proyecto de Stephen Spielberg denominado *The Minority Report*. Underkoffler finalmente se convirtió en Asesor de Ciencia y Tecnología de la película y, con Alex

McDowell, Diseñador de Producción de la misma, crearon la interfaz de usuario que Tom Cruise utiliza en la película. Algunos de los conceptos de diseño de la interfaz de usuario de *The Minority Report* finalmente terminaron en otro proyecto de Underkoffler llamado *G-Speak*.

Quizá la contribución de Underkoffler más fascinante al diseño a la película fue una sugerencia que hizo a Spielberg para que Cruise accidentalmente desordenara su escritorio virtual en una escena en la que el protagonista se vuelve y estrecha la mano de Colin Farrell. Es una escena que capta el reconocimiento desagradable de que incluso interfaces de ordenador inteligentes siguen siendo dependientes en última instancia de convenciones, y que estas convenciones son fácilmente socavadas por la realidad extraña de la vida real.

Ilustración 4 Interfaz de usuario natural representada en The Minority Report

The Minority Report fue estrenada en 2002. Imágenes de la película se filtraron de inmediato en el inconsciente de la comunidad, colgando en el espíritu de la época como una advertencia. Se empezó a sentir una leve incertidumbre sobre la prevalencia del ratón en la vida cotidiana y la prensa así como la opinión popular comenzó a hablar de lo que hemos venido a llamar la interfaz de usuario natural. Microsoft comenzó a trabajar en la plataforma *multitouch Surface* en 2003, comenzó a enseñarla en el año 2007, y, finalmente, fue lanzada en 2008. Apple mostró el iPhone al mundo en 2007. El iPad se comenzó a vender en 2010. La llegada de cada tecnología *NUI* al mercado estuvo acompañada por comparaciones con *The Minority Report*.

El efecto de The Minority Report

Tanta tinta se ha derramado sobre la influencia evidente de *The Minority Report* en el desarrollo de Kinect que es prácticamente de obligado cumplimiento que sea mencionada aquí.

Una de las críticas más peculiares a la película fue la opinión del crítico de cine Roger Ebert afirmando que ofrecía una "vista previa optimista" del futuro. *The Minority Report*, basada parcialmente en un relato corto de Philip K. Dick,

describe un futuro en que la vigilancia policial es un fenómeno generalizado hasta el punto de predecir crímenes antes de que sucedan y por tanto de encarcelar a aquellos que aún no han cometido dichos crímenes. En la sociedad se da de forma masiva y generalizada el uso de anuncios publicitarios en los que se reconoce la retina de la gente en lugares públicos para producir mensajes de marketing basados en bases de datos demográficas y se almacenan estos datos en la nube. En este futuro se muestran experimentos genéticos que producen monstruosas plantas carnívoras, arañas robots que deambulan por las calles, un próspero mercado negro de partes del cuerpo que permite a las personas cambiar sus identidades y (tal vez la predicción de futuro más irritante) el uso por parte de todos los policías de *jetpacks* o mochilas con cohetes para poder "volar".

Quizá lo que Ebert criticó fue la noción del mundo de *The Minority Report* como un futuro creíble, extrapolado de nuestro mundo, demostrando que a través de la tecnología nuestro mundo puede cambiar y no únicamente ser más de lo mismo. Incluso si introduce nuevos problemas, la ciencia ficción reafirma la idea de que la tecnología nos puede ayudar a salir de nuestros problemas actuales. En el libro de 1958 *La condición humana*, la autora y filósofa Hannah Arendt caracteriza el papel de la ciencia ficción en la sociedad diciendo:

> *"...la ciencia ha afirmado que lo que los humanos previeron en sueños no era ni salvaje ni parado...enterrado en la no muy respetada literatura de ciencia ficción (a la que, lamentablemente, hasta ahora nadie ha prestado la atención que se merece como vehículo de sentimientos y deseos de las masas)"*

Si bien no todo el mundo anhela la llegada de *jetpacks* o de arañas robóticas, la mayoría de nosotros aspiramos a que la tecnología cambie significativamente nuestras vidas en un futuro próximo.

Lo que es interesante acerca *The Minority Report* y, antes de ésta, de series de ciencia ficción como la franquicia de Star Trek, es que sólo no suelen predecir el futuro sino que incluso pueden darle forma. Ejemplos son las puertas deslizantes de cualquier comercio (como las puertas deslizantes del USS Enterprise), los móviles de tapa (el comunicador del capitán Kirk), por ejemplo.

Si *The Minority Report* impulsó el diseño y la adopción del sistema de reconocimiento de gestos en Kinect, se puede decir que Star Trek impulsó las capacidades de reconocimiento de voz de Kinect. En entrevistas con empleados y ejecutivos de Microsoft, hay repetidas referencias a la voluntad de diseñar Kinect de forma que funcione como el ordenador o el sistema holográfico del USS Enterprise. Hay una sensación en esas entrevistas de que si la parte de reconocimiento de discurso (*speech recognition)* del dispositivo no se resolviera (y en ocasiones hubo discusiones sobre olvidar esta funcionalidad, ya que se retrasó), el sensor de Kinect no habría sido el sensor futurista que todo el mundo esperaba.

El proyecto secreto de Microsoft

En el mundo de los videojuegos, Nintendo dio la sorpresa en el *Tokyo Game Show* de 2005 con la presentación de la consola Wii. La consola fue acompañada por un nuevo dispositivo de juego llamado *Wiimote*. Al igual que los cubos magnéticos del proyecto original *Put-That-There*, el controlador de Wii puede detectar el movimiento a lo largo de los tres ejes. Además, este controlador contiene un sensor óptico que detecta dónde está apuntando. Está alimentado por batería, eliminando las clásicas cuerdas de consolas de otras plataformas.

Más tarde, en 2006, Peter Moore, el entonces jefe de la división Xbox de Microsoft, comenzó a trabajar en una alternativa competitiva a Wii. También fue en ese momento cuando Alex Kipman, director de un equipo de incubación de ideas dentro de la división de Xbox, se reunió con los fundadores de PrimeSense en la *Electronic Entertainment Expo* (*E3*) de 2006. Microsoft creó dos equipos que compitieron para desarrollar la alternativa a Wii: un equipo trabajaría con la tecnología PrimeSense y el otro con tecnología desarrollada por la compañía 3DV. Aunque el objetivo original era dar a conocer alguna novedad en el E3 de 2007, ninguno de los equipos tenían nada lo suficientemente

pulido como para presentar para la fecha de la exposición. Este proceso fue dejado al margen en 2007 cuando Peter Moore anunció que dejaba Microsoft para ir a trabajar a Electronic Arts.

En el verano de 2007 el trabajo secreto que el equipo de Xbox está haciendo estaba ganando impulso internamente en Microsoft. En la conferencia *All Things Digital* de ese año, Bill Gates fue entrevistado junto a Steve Jobs. Durante la entrevista, en respuesta a una pregunta sobre Microsoft Surface, y si las interfaces *multitouch* se convertirían en una corriente principal dentro de Microsoft, Gates empezó a hablar del reconocimiento por visión como un paso más allá del *multitouch*:

> ***Gates**: El software está empezando a usar la vista. De esta forma, imagine una máquina de juego en el que usted puede simplemente coger el bate de baseball y agitarlo enfrente del televisor, o con una raqueta de tenis.*
>
> ***Entrevistador**: Ya existe una. Se llama Wii.*
>
> ***Gates**: No. No. No es eso. Usted no puede coger su raqueta de tenis usarla directamente. No puede sentarse con sus amigos y hacer esas acciones naturales. Wii es un dispositivo de posición 3D. Esto es reconocimiento de vídeo. Se trata de una cámara que ve lo que está pasando. En una reunión, cuando se está en una conferencia, usted no sabe quién está hablando cuando se trata sólo de audio... con la cámara esto será ubicuo... el software podrá usar la vista, y podrá hacerlo de forma muy, muy barata… y eso significa que este material estará en todas partes. En vez de definirlo como un dispositivo portátil, se trataría de una parte de la sala de reunión o la sala de estar...*

Sorprendentemente el entrevistador interrumpió a Gates durante su exposición sobre el futuro de la tecnología y la conversación siguió de nuevo a lo más importante en 2007: ordenadores portátiles. Sin embargo, Gates reveló en esta entrevista que Microsoft ya estaba pensando en la nueva tecnología desarrollada por el equipo de Xbox como algo más que un mero dispositivo de juegos. Se pensaba ya además como un dispositivo para la oficina.

Después de que Moore saliera de Microsoft, Don Matrick tomó las riendas guiando el equipo de Xbox. En 2008, retomó el proyecto secreto de reconocimiento de video en torno a la tecnología de PrimeSense. Si bien parece que la tecnología de 3DV nunca consiguió llegar a implementarse en Kinect, Microsoft compró la compañía en 2009 por 35 millones de dólares. Esto se hizo al parecer con el fin de defenderse contra posibles litigios de patentes en torno a Kinect. Alex Kipman, gerente de Microsoft desde el año 2001, fue ascendido a director general de proyectos de incubación y puesto al cargo de crear el nuevo dispositivo con nombre en clave *Project Natal* que debía incluir un sensor de profundidad, seguimiento de movimiento, reconocimiento facial y reconocimiento de voz.

Curiosidad: Tradicionalmente Microsoft nombrado a grandes proyectos usando como código nombres de ciudades. Alex Kipman nombró el proyecto secreto de Xbox *Natal*, en honor a su ciudad de origen en Brasil.

El dispositivo de referencia creado por PrimeSense incluía una cámara RGB, un sensor de infrarrojos, y un emisor de infrarrojos. Microsoft licenció el diseño de referencia de PrimeSense así como el diseño del chip PS1080, que era capaz de procesar datos de profundidad a 30 imágenes por segundo. Lo más importante es que era capaz de procesar los datos de profundidad de una manera innovadora que redujo drásticamente el precio del reconocimiento de la profundidad en comparación con el método vigente en el momento denominado *tiempo de vuelo*, una técnica que registra el tiempo que tarda un rayo de luz en salir y volver al sensor. La solución de PrimeSense fue proyectar un patrón de puntos infrarrojos a través de la superficie de las habitaciones y usar el espaciamiento entre los puntos para formar un mapa de profundidad de 320X240 píxeles analizado por el chip PS1080. El chip también alineaba automáticamente la información de la cámara RGB y la cámara de infrarrojos, proporcionando datos RGBD.

Microsoft añadió una matriz de micrófonos de cuatro piezas a esta estructura básica, proporcionando eficazmente un micrófono direccional para reconocimiento de voz que sería efectivo en una gran habitación. Microsoft ya tenía años de experiencia con el reconocimiento de voz, que ha estado disponible en su sistema operativo Windows desde XP.

Kudo Tsunada, proveniente de Electronic Arts se añadió al equipo del proyecto, liderando su propio equipo de incubación para crear prototipos de juegos para el nuevo dispositivo. Él y Kipman tenía una fecha límite del 18 de agosto de 2008 para mostrar a un grupo de ejecutivos de Microsoft lo que *Project Natal* podía hacer. El equipo de Tsunada consiguió realizar 70 prototipos, algunos de los cuales fueron mostrados a los ejecutivos. El proyecto obtuvo entonces luz verde y el verdadero trabajo comenzó. Se les dio una fecha de lanzamiento de *Project Natal*: Navidad de 2010.

Microsoft Research

Si bien el problema de hardware se resolvió sobre todo gracias a PrimeSense (todo lo que quedaba era dar al dispositivo una forma más pequeña), los problemas de software parecían insuperables. En primer lugar, se tuvo que crear un sistema de reconocimiento de movimiento basado los flujos de datos de la cámara RGB y del sensor de profundidad del dispositivo. A continuación, se realizó un gran proceso de depuración con el fin de hacer que la señal de audio fuese viable para la plataforma de reconocimiento de voz. El equipo de *Project Natal* recurrió a Microsoft Research (MSR) para resolver éstos problemas.

Ilustración 5 Presentación de Project Natal

MSR es una inversión anual de millones de dólares por parte de Microsoft. Las diferentes instalaciones de MSR se dedican típicamente a la investigación pura en ciencias de la computación y la ingeniería en lugar de tratar de crear productos para la marca padre. Debió de resultar extraño cuando el equipo de Xbox recurrió a diferentes secciones de Microsoft Research no sólo para que les ayudaran a llegar a conseguir diseñar el producto, sino para que éste se realizara siguiendo un ciclo de producto muy corto.

In a finales de 2008, el equipo de *Project Natal* contactó con Jamie Shotton en la oficina del MSR en Cambridge, Inglaterra, para ayudar con el problema del seguimiento de movimiento. La solución para el seguimiento de movimiento que el equipo de Kipman había ideado contaba con varios problemas. En primer lugar, se basaba en que el jugador debía colocarse en una primera posición en forma de T para permitir al software de captura de movimiento descubrirlo. Luego, de vez en cuando perdía al jugador durante el movimiento, obligando al jugador a reiniciar el

sistema y una vez más asumir la pose en T. Por último, el software de seguimiento de movimiento sólo funcionaba para el tipo de cuerpo que fue diseñado: el de los ejecutivos de Microsoft.

Por otra parte, los datos de profundidad proporcionados por el sensor resolvían varios problemas principales acerca del seguimiento de movimiento. Los datos de profundidad permiten filtrar fácilmente los píxeles que no pertenecen al jugador. Información extra como el color y la textura de la ropa de los jugadores también se filtran mediante los datos de profundidad. Lo que queda es básicamente una representación del jugador como un conjunto representado en posiciones de pixel (*player blob*), tal como se muestra en la Figura 1. Los datos Los datos de la cámara de profundidad además proporcionan información sobre la altura y la anchura del jugador en metros.

El reto para Shotton fue convertir este esquema de una persona en algo que pudiera ser "seguido". El problema, como él lo veía, era dividir este *player blob* obtenido por el flujo de profundidad en partes reconocibles del cuerpo. A partir de estas partes del cuerpo, las articulaciones podrían ser identificadas, y a partir de estas articulaciones, el esqueleto podría ser reconstruido. Trabajando con Andrew Fitzgibbon y Andrew Blake, Shotton creó un algoritmo que podía distinguir a 31 partes del cuerpo (ver Figura 1-2). A partir de estas partes, la versión de Kinect mostrada en el E3 en 2009 pudo producir 48 articulaciones (el SDK de Kinect, por el contrario, expone 20 articulaciones).

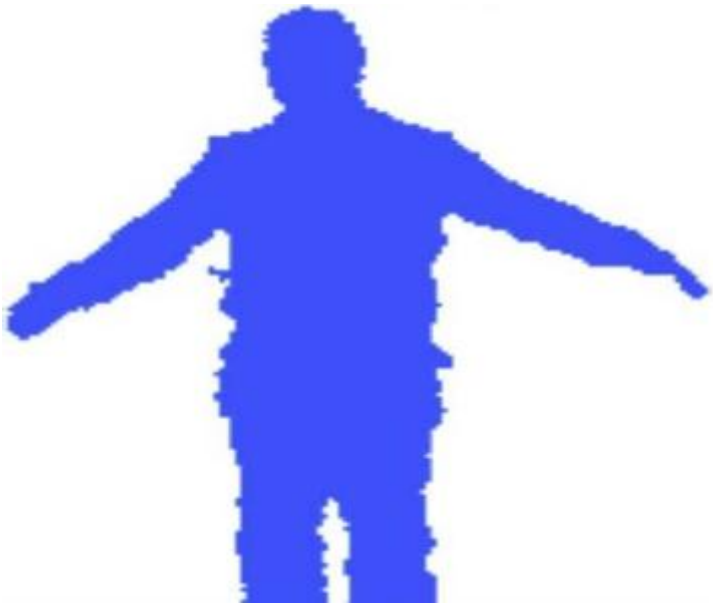


Ilustración 6 Player blob

Ilustración 7 Partes del cuerpo del jugador.

Para resolver el problema de la posición en T inicial requerida al jugador para la calibración, Shotton decidió recurrir a técnicas de aprendizaje por ordenador. Con montones y montones de datos, el software de reconocimiento de imagen podría ser entrenado para dividir el *player blob* en partes del cuerpo que pudieran utilizarse. Se enviaron equipos a hogares para realizar grabaciones de vídeo de personas realizando movimientos físicos básicos. Se recogieron datos adicionales en un estudio de captura de movimiento en Hollywood de personas bailando, corriendo, y realizando acrobacias. Todos este material de vídeo fue entonces enviado aun sistema de computación distribuida llamado *Dryad* que había sido desarrollado por otra rama de investigación de Microsoft en Mountain View, California, con el fin de comenzar a generar un clasificador por árbol de decisión para poder asignar cualquier píxel del flujo de datos de profundidad de Kinect en una de las 31 partes del cuerpo. Este proceso se realizó para 12 tipos diferentes de cuerpos y fue en repetidas ocasiones ajustado para mejorar la capacidad del software de decisión para identificar una persona sin una pose inicial, sin interrupciones en el reconocimiento, y para diferentes estructuras fisiológicas de personas (forma de cuerpo).

Con lo anterior se abarcan los aspectos de Kinect relacionados con *The Minority Report*. Para manejar la parte de Star Trek, Alex Kipman recurrió a Iván Tashev del grupo de investigación de Microsoft en Redmond. Tashev y su equipo habían trabajado previamente en la implementación de matrices o arrays de micrófonos en Windows Vista. Así como la necesidad de que el sensor fuese capaz de filtrar los píxeles que no pertenecieran a jugadores era parte clave de la solución al problema de reconocimiento de esqueleto, el proceso de filtrado de ruido de fondo en una matriz de micrófonos situada mucho más cerca de un sistema estéreo (altavoces del televisor) que del usuario que habla fue el principal problema a la hora de diseñar el sistema reconocimiento de voz de Kinect. Usando una combinación de tecnologías patentadas (incluidas de forma gratuita en el SDK de Kinect para Windows), el equipo de Tashev obtuvo unos algoritmos para la cancelación de ruido y la supresión de eco que mejoraron el procesamiento de audio situándolo muy por encima del estándar que estaba disponible en la fecha.

Basado en esta depuración de audio, un sistema de aprendizaje computacional distribuido de mil nodos se dedicó durante una semana a la construcción de un modelo acústico para Kinect en base a varios acentos regionales de América y las peculiares propiedades acústicas de la matriz de micrófono de Kinect. Este modelo se convirtió en la

base de la función *TellMe* incluida con la consola Xbox, así como del paquete de idioma del tiempo de ejecución de Kinect usado en el SDK. El modelo acústico no fue completamente finalizado hasta el 26 de septiembre 2010. Poco después, el 4 de noviembre, el sensor de Kinect fue anunciado.

La carrera para hackear Kinect

El lanzamiento de Kinect fue recibido con numerosas críticas. Webs especializadas en jugos reconocieron en general que la tecnología era muy interesante pero que cansaba a los jugadores rápidamente. Esto no disminuyó las ventas de Kinect. Espacio el dispositivo vendió de media 133 mil unidades al día durante los primeros 60 días de lanzamiento, rompiendo los récords de ventas de a iPhone e iPad y estableciendo un nuevo récord Guinness mundial. No es que las webs de análisis de juegos estuvieran equivocadas respecto al factor novedoso de Kinect, sino que la gente estaba esperando Kinect de todas formas aunque el jugar en sólo una hora al día o varias horas. Era una parte del futuro que ellos querían tener en su cuarto de estar.

Ilustración 8 Open Kinect

La excitación del mercado general se emparejó con la excitación de la comunidad hacker. La historia *hacker* empieza con Johnny Chung Lee, el hombre que originalmente había *hackeado* el controlador remoto de Wii para implementar el seguimiento de dedos y que más tarde fue contratado para el proyecto natal para trabajar en el reconocimiento de gestos. Frustrado por el fallo en los esfuerzos de Microsoft de publicar un driver público, Lee contacto AdaFruit, un suministrador de código libre, para acoger un concurso con el fin de hackear Kinect. El concurso, anunciado el día de lanzamiento de Kinect, fue construido alrededor de una interesante característica de hardware del sensor Kinect: utiliza un conector USB estándar para comunicarse con la Xbox. Este mismo conector USB puede ser conectado en el puerto USB de cualquier PC o portátil. La primera persona que creara con éxito un controlador para el dispositivo y escribir una aplicación de conversión de los flujos de datos del sensor en vídeo y visión de profundidad ganaría la recompensa de 1.000 dólares que Lee había ofrecido para la prueba.

El mismo día, Microsoft hizo la siguiente declaración en respuesta al concurso de AdaFruit: "Microsoft no aprueba la modificación de sus productos... Con Kinect, Microsoft construyó el hardware con numerosas salvaguardias de software diseñadas para reducir las posibilidades de modificación. Microsoft continuará haciendo avances en este tipo de garantías y trabajar estrechamente con las fuerzas del orden y los grupos de seguridad de productos para

mantener Kinect a prueba de manipulaciones". Lee y los responsables de AdaFruit aumentaron la recompensa hasta 2.000 dólares.

Para el 6 de noviembre de 2010, Joshua Blake, Sandler Seth, y Machulis Kyle y otros habían creado el grupo de trabajo OpenKinect para ayudar a coordinar esfuerzos en torno al concurso. Su idea era que el problema del controlador era solucionable, pero que la duración de los esfuerzos para hackear Kinect para PC implicaría el intercambio de información y la construcción de herramientas basadas en la tecnología. Ya estaban mirando más allá del concurso de AdaFruit e imaginando lo que vendría después. El 7 de noviembre se decidió que la recompensa se repartiría entre la comunidad OpenKinect si alguien ganaba el concurso, para olvidar la faceta del premio y mirar hacia lo que podría hacerse con la tecnología de Kinect. Este grupo de trabajo fue el hogar de la comunidad de hackers de Kinect para el siguiente año.

Simultáneamente el 6 de noviembre, un hacker conocido como AlexP fue capaz de controlar los motores de Kinect y leer sus datos del acelerómetro. La recompensa de AdaFruit se elevó a 3.000 dólares. El lunes 8 de noviembre AlexP compartió un vídeo en el que se muestra cómo obtiene datos del flujo de RGB y flujos de datos de profundidad del sensor de Kinect y mostrarlos. Sin embargo, no pudo recoger el premio, debido sus reservas sobre hacer su código público. Ese mismo día, Microsoft aclaró su posición anterior de una manera que parecía permitir los esfuerzos en curso para hackear Kinect, siempre y cuando no se llamara al proceso “hackear”:

> *“Kinect para Xbox 360 no ha sido hackeado de forma alguna, ya que tanto el software como el hardware que forman parte de Kinect para Xbox 360 no se han modificado. Lo que ha ocurrido es que alguien ha creado controladores que permiten a otros dispositivos interactuar con Kinect para Xbox 360. La creación de estos controladores, y el uso de Kinect para Xbox 360 con otros dispositivos, no están soportados. Animamos a los clientes a utilizar Kinect para Xbox 360 con su Xbox 360 para obtener la mejor experiencia posible.”*

El 9 de noviembre, AdaFruit finalmente recibió un analizador de USB, el Beagle 480, y se puso a trabajar en la publicación de volcados USB de datos provenientes del sensor Kinect. La comunidad Open Kinect, que se hacía llamar "Tiger Team", comenzó a trabajar en estos datos a través de un canal de IRC e hicieron grandes avances en cuestión de días. Al mismo tiempo, sin embargo, Héctor Martín, el ingeniero informático de Bilbao, España, acababa de comprarse un sensor Kinect y había comenzado a analizar los datos de AdaFruit. A las pocas horas había escrito el controlador y la aplicación para mostrar datos RGB y de profundidad. El premio AdaFruit fue reclamado por tanto en tan sólo siete días.

Martin se convirtió en colaborador del grupo de OpenKinect y una nueva biblioteca, *libfreenect*, se convirtió en la herramienta base de los esfuerzos de hackers de la comunidad. Joshua Blake anunció la contribución de Martin a OpenKinect en la siguiente nota:

> *Me quedé sorprendido con Héctor justo después de que publicó el video en el IRC y hable con él acerca de este grupo. Él dijo que estaría encantado de unirse a nosotros (y de hecho ya se ha suscrito). Después de que duerma para recuperarse, vamos a hablar un poco más sobre la integración de su trabajo y nuestro trabajo.*

Ilustración 9: Analizador de USB Beagle 480

A partir de aquí es cuando la verdadera diversión comenzó. A lo largo de noviembre, la gente comenzó a publicar vídeos en la Web mostrando lo que podían hacer con Kinect. Manifestaciones artísticas basadas en Kinect, experiencias de realidad aumentada o experimentos de robótica empezaron a aparecer en YouTube. Sitios como KinectHacks.net florecieron para hacer un seguimiento las cosas que la gente estaba construyendo con Kinect. El 20 de noviembre, alguien había publicado un video de un simulador de sable de luz usando Kinect (otra aspiración de película cumplida). Microsoft, mientras tanto, no estaba ociosa. La empresa observó con entusiasmo, mientras cientos de *hacks* de Kinect florecían en la Web.

El 10 de diciembre, PrimeSense anunció el lanzamiento de sus propios controladores de código abierto para Kinect, junto con las bibliotecas necesarias para trabajar con los datos. Esto proporcionó mejoras en los algoritmos de seguimiento de esqueleto que mejoraban lo que era posible hasta entonces con *libfreenect,* haciendo que proyectos que requerían integración entre datos RGB y de profundidad comenzaran a migrar de OpenNI a PrimeSense. Sin embargo, sin los detalles de la tecnología diseñada por Microsoft Research, el seguimiento de esqueleto con OpenNI todavía requería la incómoda postura en T para inicializar el reconocimiento de esqueleto.

El 17 de junio de 2011, Microsoft finalmente lanzó la versión beta del SDK para Kinect al público bajo una licencia no comercial, después de mostrarlo durante varias semanas en eventos como el MIX (conferencia de Microsoft sobre nuevas tendencias). Según lo prometido, incluía los algoritmos de reconocimiento de esqueleto que evitaban realizar una pose inicial, así como la tecnología AEC (*Acoustic Echo Cancellation* para cancelación de eco) y modelos acústicos necesarios para usar el reconocimiento del habla con Kinect en una gran sala.

En la actualidad, todos los desarrolladores tienen acceso a las mismas herramientas que Microsoft utiliza internamente para el desarrollo de aplicaciones Kinect.

2.3. Lanzamiento

Microsoft asignó un presupuesto de publicidad en Estados Unidos de 500 millones de dólares para el lanzamiento de Kinect, una suma mayor que la inversión en el lanzamiento de la consola Xbox. Los planes incluyeron anunciar a Kinect en medios públicos de la Web como YouTube, anuncios en cadenas de televisión como *Disney* o *Nickelodeon*, así como en series de televisión como *K* y *Glee*. Los anuncios impresos se publicaron en la revistas como *People* o *InStyle*, mientras que marcas como *Pepsi*, *Kellogg's* y *Burger King* también llevarán anuncios de Kinect. También tuvo especial relevancia un evento publicitario de Kinect organizado en Times Square, en la ciudad de Nueva York.

El 19 de octubre, antes del lanzamiento Kinect, Microsoft anuncia Kinect en *The Oprah Winfrey Show* dando gratis una Xbox 360 y un Kinect para la gente que estaba en el público. Más tarde, también regaló Kinect con Xbox 360 para el público de *The Ellen Show* y *Late Night With Jimmy Fallon*.

El 23 de octubre, Microsoft celebró una fiesta de prelanzamiento para Kinect en Beverly Hills. Los huéspedes fueron invitados a las sesiones de prueba en las que se enseñó el nuevo sensor con los juegos *Dance Central* y *Kinect Adventures*.

Ilustración 10 Publicidad de Kinect para Xbox 360

2.4. Tecnología

El sensor Kinect es básicamente un dispositivo que contiene dos cámaras y un array de micrófonos.

Ilustración 11 Sensor Kinect, vista frontal

Kinect se basa en tecnología de software desarrollada por Rare, una subsidiaria de Microsoft Game Studios propiedad de Microsoft, y en la tecnología de cámara de profundidad de la desarrolladora israelí PrimeSense, que desarrolló un sistema que podía interpretar gestos específicos, haciendo posible el control de dispositivos electrónicos sin el uso de las manos mediante el uso de un proyector de infrarrojos, una cámara y un microchip especial para seguir el movimiento de los objetos y personas en tres dimensiones. Este sistema de escáner 3D llamado *Light Coding* emplea una variante de la reconstrucción 3D basada en imagen.

El sensor Kinect está diseñado para ser colocado longitudinalmente por encima o por debajo de la pantalla de vídeo para usar junto con la videoconsola Xbox 360. El dispositivo cuenta con una cámara RGB, un sensor de profundidad, un micrófono de múltiples matrices y un procesador personalizado que ejecuta el software patentado, que proporciona captura de movimiento de todo el cuerpo en 3D, reconocimiento facial y capacidades de reconocimiento de voz. El micrófono de matrices del sensor de Kinect permite a la Xbox 360 llevar a cabo la localización de la fuente acústica y la supresión del ruido ambiente.

En el lanzamiento, el reconocimiento de voz se hizo sólo disponible en Japón, el Reino Unido, Canadá y los Estados Unidos. Europa continental recibió la función más tarde en la primavera de 2011. En la actualidad el reconocimiento de voz es compatible en Australia, Canadá, Francia, Alemania, Irlanda, Italia, Japón, México, Nueva Zelanda, Reino Unido y Estados Unidos. El array de micrófonos del sensor Kinect permite calcular la localización de una fuente acústica y la supresión del ruido ambiente. Ésta función de localización de fuente acústica será la más importante para los intereses de este documento.

Especificaciones del dispositivo

Elemento	Rango
Ángulo de visión	43° vertical y 57° horizontal
Rango de movimiento del cabezal (vertical)	±27°
Velocidad de imagen (flujos de datos de color y profundidad)	30 frames per second (FPS)
Resolución por defecto en cámara de profundidad	VGA (640 x 480)

Resolución por defecto en cámara de color RGB	VGA (640 x 480)
Formato de audio	16-kHz, 16-bit mono pulse code modulation (PCM)
Características de entrada de audio	Un array de cuatro micrófonos que incluye conversos analógico a digital de 24 bits integrado con Acoustic Echo Cancellation y Noise Suppression

Sensor de profundidad

El sensor de profundidad se compone de un proyector láser de infrarrojos combinado con un sensor CMOS monocromo, que captura los datos de vídeo en 3D en cualquier condición de luz ambiental. El rango de detección del sensor de profundidad es ajustable, y el software a través de las APIs es capaz de calibrar automáticamente el sensor basado en el juego y el ambiente físico del usuario, con capacidad para detectar la presencia de los muebles u otros obstáculos.

Descrito por el personal de Microsoft como la principal innovación de Kinect, la tecnología de software implementada en el sensor permite un avanzado reconocimiento de gestos, reconocimiento facial y reconocimiento de voz. Según la información facilitada por Microsoft, Kinect es capaz al mismo tiempo de realizar el seguimiento de hasta seis personas, de las cuales dos podrían ser activos para el análisis de movimiento con una extracción de características de 20 puntos característicos del usuario.

Sistema de rastreo

El sistema óptico de Kinect es lo que permite seguir los movimientos en tiempo real de una persona (está exclusivamente orientado a reconocer figuras humanas). Es sistema es complejo y está formado por la integración de tecnología que lleva desarrollándose desde hace 15 años junto con efectos y funcionalidades que se han descubierto recientemente, con fin de obtener un costo mucho menor a los miles de dólares que costaba hace algunos años realizar mediante cámaras de tiempo de vuelo y similares.

El sistema está conformado por dos partes: un proyector y una cámara VGA infrarroja, los cuales funcionan proyectando un láser a través de toda el área donde se encuentra el dispositivo generando un denominado *depth field* o campo de profundidad, que abarca el área cubierta por el ángulo de visión del sensor. Esencialmente, todos los pixeles que Kinect recibe como ruido de IR son asignados un valor dentro del rango de 11 bits que maneja el sensor. Por lo tanto, Kinect devuelve una matriz que representa una imagen capturada por el sensor de profundidad. La diferencia con la cámara RGB (y con las cámaras RGB estándar) es que cada valor de esa matriz no representa un color, sino el valor de la distancia desde el sensor hasta ese punto en el espacio de visión. Por lo tanto, es posible tomar estos datos y representar los elementos situados en el área de visión del sensor y representarlos usando, por ejemplo, distintos colores.

Una vez capturada una imagen, el software la hace pasar por una serie de filtros en los que se determina si hay presentes seres humanos en la escena o no (es decir, separar lo que es una persona y que no lo es). Para esto se utilizan una serie de parámetros cargados en el sensor y que representa la fisiología del ser humano, que permiten diferenciar entre otros elementos presentes en la escena como mobiliario, animales, etc. que impiden que la mesa del café o el perro del vecino sean reconocidos como otros usuarios.

Una vez que la información sobre personas es separada del resto, se convierte cada identificación de cuerpo en un esqueleto con 20 articulaciones móviles. Las manos se toman como un objeto en lugar de capturar los dedos, así que no es posible detectar el movimiento de los dedos de la mano usando Kinect.

Ilustración 12 Detalle de cámara RGB y sensor de profundidad (emisor y receptor infrarrojo).

Hasta la versión 1.0 del SDK, el sensor sólo podía detectar el esqueleto de jugadores/usuarios que se encuentren de pie, no sentados, y el reconocimiento de esqueleto en 20 puntos no está disponible en el Modo Cercano.

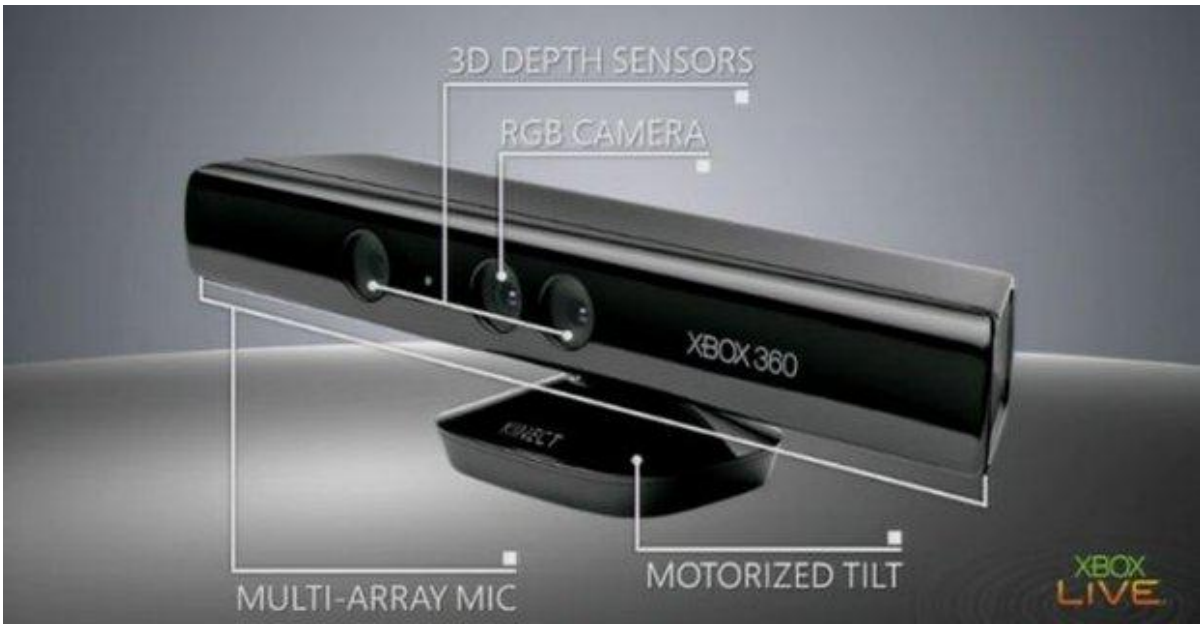


Ilustración 13 Componentes del sensor Kinect

Reconocimiento de voz

El array de micrófonos de Kinect cuenta con cuatro micrófonos y cada uno funciona en un canal de audio de 16 bits a una velocidad de muestreo de 16 kHz.

Ilustración 14 Estructura interna de Kinect

El problema principal al que se enfrenta el subsistema de micrófonos, es que necesita capturar e interpretar sonidos provenientes del usuario de la videoconsola (aproximadamente 3 metros de distancia), al mismo tiempo que debe ignorar ruidos ambientales y otros sonidos que no sea la voz del usuario. Para resolver esta situación, el laboratorio de Microsoft Research analizó la situación en 250 residencias con 16 micrófonos y tomó grabaciones de distintos ambientes, determinando al final la mejor configuración posible.

El resultado fue un array de micrófonos localizados en las laterales del dispositivo, específicamente uno a la izquierda y tres a la derecha del centro. De hecho, ésta es la razón por la que el Kinect es tan ancho.

El procesador de audio es capaz de cancelar los ruidos que provienen ruidos ambientales provenientes del entorno (televisor, sistema de sonido, etc.) así como de calcular el origen de una fuente acústica (técnica más conocida del inglés *beamforming*).

Kinect tiene además un modelo acústico para cada país y dialectos para regiones específicas, construido gracias a cientos de actores alrededor del mundo, cuyas voces fueron procesadas al hablar de varias formas.

Motor

El sensor contiene un mecanismo de inclinación motorizado y en caso de usar un Xbox 360 del modelo original, tiene que ser conectado a una toma de corriente, ya que la corriente que puede proveerle el cable USB es insuficiente; para el caso del modelo de Xbox 360 S esto no es necesario ya que esta consola cuenta con una toma especialmente

diseñada para conectar el Kinect y esto permite proporcionar la corriente necesaria que requiere el dispositivo para funcionar correctamente.

Este motor es capaz de inclinar el dispositivo en un ángulo de 30 por arriba y por abajo del plano horizontal.

Rango de operación

El sensor tiene un campo angular de visión de 57 ° en sentido horizontal y 43 ° verticalmente, mientras que el pivote motorizado es capaz de inclinar el sensor hasta 27 ° ya sea hacia arriba o hacia abajo. En la siguiente imagen vemos los rangos de distancias (en metros) entre los que Kinect puede operar, según Microsoft:

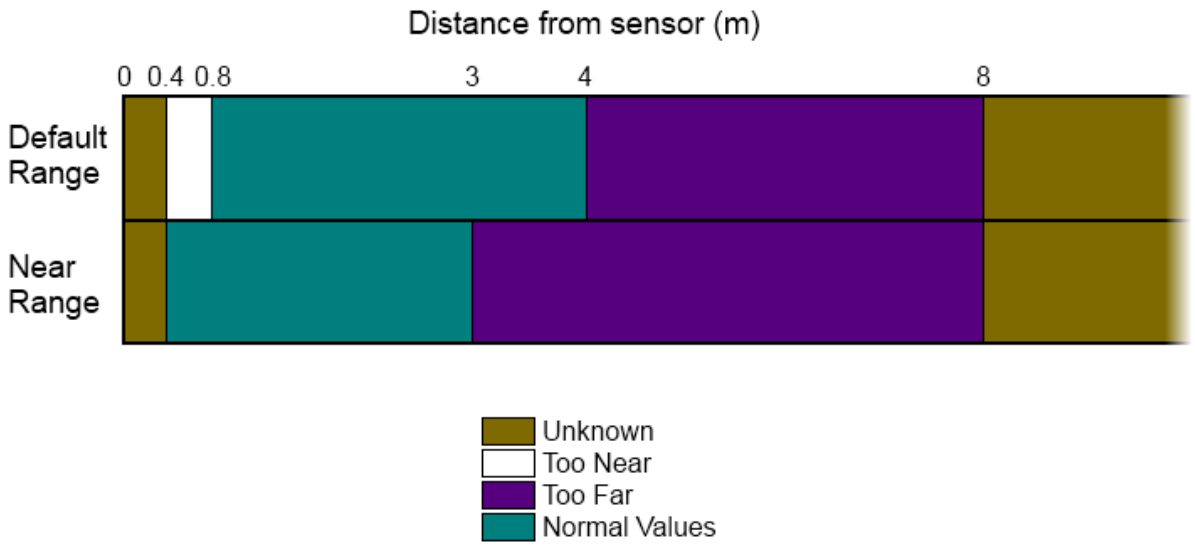


Ilustración 15 Rango de operación del sensor Kinect

El rango por defecto (*Default Range*) se refiere a la versión del sensor para Xbox 360 y para Windows, mientras que el Modo Cercano (*Near Range*) se refiere exclusivamente al sensor para Windows.

2.5. Recepción por el público

Kinect ha obtenido críticas mayoritariamente positivas; el sitio web *IGN* le dio un puntaje de 7,5 sobre 10 y comentó que "puede significar una enorme cantidad de diversión para los jugadores casuales, y el concepto creativo y libre de controladores resulta innegablemente atractivo", aunque percibió que por "149,99 dólares americanos, un accesorio que es básicamente una cámara que detecta el movimiento para la Xbox 360 es una adquisición delicada, considerando sobre todo si se lo compara con el precio de la Xbox 360, la cual cuesta 199,99 por sí sola". *Game Informer* le brindó una calificación de 8 de 10 y, aunque elogió la tecnología, percibió que se requiere algo de tiempo para acostumbrarse a su uso, además de que el requerimiento espacial podría plantear un obstáculo. *Computer and Video Games* lo catalogó como una joya tecnológica, elogiando los controles de voz y gestos, sin embargo criticó la línea de juegos disponible al momento de su lanzamiento así como el Kinect Hub, una versión simplificada de opciones en pantalla diseñada para el sensor de Kinect y cuya función es facilitar la navegación en el menú principal. Al final, le dio un puntaje de 8,8 de 10.11 A su vez, *CNET* le dio una calificación de 3,5 de 5, al considerar que el dispositivo mantiene activos a los jugadores, y es difícil de engañar pues cuenta con un sensor que detecta el movimiento del cuerpo entero, aunque calificó negativamente algunos aspectos como la sincronización de la curva de aprendizaje, la fuente adicional de alimentación requerida para las versiones más antiguas de la Xbox 360, los requerimientos espaciales así como los requisitos de duración para varios juegos, además de percibir que la sensación de novedad

inherente en la navegación del menú y el video desaparece al poco tiempo de usar el accesorio. *Engadget* le brindó una calificación baja de 6 sobre 10, pues aunque elogió la tecnología y el potencial de sus juegos de yoga y baile, criticó el gran requerimiento de espacio, la lentitud de la interfaz de usuario en cuanto al movimiento de las manos y su línea primeriza de juegos. *Kotaku* mencionó que Kinect se sintió revolucionario la primera vez que se usó, sin embargo a veces los juegos son incapaces de reconocer algunos movimientos o son lentos para asimilarlos. El blog citado subrayó que la experiencia de usuario es inconsistente si se toma en cuenta la ausencia de una aplicación o juego asesinos, además de comentar que el accesorio tiene que ser algo más que un simple remplazo de controles remotos para valer su precio: "no es aún algo del todo imprescindible [el Kinect], sino más bien como algo que eventualmente podría ser imprescindible". El sitio 1UP.com le dio una evaluación de B-, comentando que aunque "Kinect es una gran novedad para las reuniones familiares y entre amigos que no son típicamente jugadores, se requiere de bastante espacio para poder usarlo, y la mayoría de la gente no cuenta con un espacio muy amplio frente a su televisor como para poder jugar adecuadamente con dicha cámara".

Desde el punto de vista del diseño técnico*, Ars Technica* expresó preocupación en cuanto a que el elemento central de Kinect, esto es su falta de controladores, podría obstaculizar el desarrollo de los videojuegos más allá de aquellos que tengan ya sea jugadores estacionarios, o que controlen el movimiento del jugador de forma automática; en su reseña, se enfocó en la similitud de los géneros de los juegos debutantes incluidos con el hardware, diciendo que éstos no eran capaces de «mostrar un mayor dominio»; así, *Ars Technica* predijo que Microsoft necesitaría lanzar eventualmente un controlador «tipo PlayStation Move» con tal de superar esta limitación.

La prensa estadounidense también evaluó positivamente a Kinect; *USA Today* la comparó con el sistema de controles futurista visto en la película Minority Report (2002), además de comentar que «jugar juegos con el accesorio se siente genial», evaluándolo con un puntaje de 3,5 de 4 estrellas. David Pogue, deThe New York Times, predijo que todo usuario sentirá "un ímpetu loco, mágico y maravilloso la primera vez que utilice el Kinect". A pesar de considerar que la lectura de movimientos resulta menos precisa que la implementación de la Wii (Wii MotionPlus), *Pogue* concluyó en su reseña que «la tecnología impresionante de Kinect crea una actividad completamente nueva que resulta ser social, para todas las edades e incluso atlética». *The Globe and Mail* lo catalogó incluso como «un nuevo producto estándar para el control basado en el movimiento»; para la publicación mencionada, el ligero retardo que posee el accesorio para detectar un movimiento físico no fue considerado como un obstáculo notable al jugar la mayoría de los títulos disponibles, y concluyó que Kinect es "un buen producto innovador", dándole un puntaje final de 3,5 de 4 estrellas. 2,5 millones de unidades del sensor Kinect se han vendido al 29 de noviembre de 2010, de los cuales 1.000.000 fueron vendidas en los 10 días de su lanzamiento en América del Norte.

2.6. Desarrollo de aplicaciones alternativas

Desde que la primera versión salió a la venta en 2010, el dispositivo Microsoft Kinect ha tenido una importante llegada al mercado, no sólo en el terreno de su consola Xbox 360 sino también con aplicaciones varias para PC, que eran desarrolladas de manera no oficial por terceros. Microsoft ha querido facilitar dicha tarea y que sean los desarrolladores los que doten de posibilidades sin límite a Kinect.

Ilustración 16 Kinect for Xbox 360 (2010)

Kinect es uno de los dispositivos más versátiles del mercado sobre todo si tenemos en cuenta que fue concebido originalmente para llevar al usuario a una nueva experiencia de juego. Precisamente, creo que ahí radica la importancia de este dispositivo, haber trascendido el objetivo para el que fue diseñado para convertirse en una plataforma de visión artificial que se utiliza en el MIT Media Lab y otros muchos centros de investigación.

Poco después de que saliera a la venta la primera versión del sensor Kinect, usuarios de todo el mundo comenzaron a modificar su uso y crear proyectos que trascendían la videoconsola, usando para ello librerías de código abierto. Por ello era sólo cuestión de tiempo que el Microsoft lanzase un conjunto de librerías y herramientas propias para para desarrollar aplicaciones sobre el sensor.

En palabras de Microsoft:

> *"We've seen a tremendous amount of interest in the Kinect for Windows commercial program, with more than 300 companies participating in our early adopter program. Although the commercial hardware and software were not released until February 1st, we're already seeing innovative examples of commercial uses from some of our early adopter participants. The possibilities are endless. In coming months, we will feature many of these examples of Kinect for Windows applications on this website as well as our blog."*

En este apartado vamos a analizar algunos proyectos interesantes que se han realizado usando el sensor y que muestran la gran aceptación que está teniendo tanto el SDK como el sensor en el público general. Estos ejemplos que se muestran a continuación han sido desarrollados por usuarios apasionados, sin ánimo de lucro y que evidencian el gran potencial de Kinect.

El principal atractivo de Kinect es su cámara de profundidad por infrarrojos, en la que se basan aplicaciones de reconocimiento de gestos de usuario. Ya que este documento está enfocado a las capacidades que ofrece Kinect para el tratamiento de audio, se ha decidido incluir en este apartado ejemplos de Kinect basados en tratamiento de vídeo con el objetivo de ofrecer una visión completa de las capacidades del sensor.

Marketing y Publicidad: Winsense

WinSense une y pone en práctica los conceptos de Marketing Interactivo e Interfaz de Usuario Natural mediante el uso de la plataforma Kinect. *WinSense* tiene la capacidad de reconocer 5 movimientos diferentes a fin de reconocer la interacción del usuario. Estos movimientos se reconocerán como acciones de usuario tales como desplazar hacia izquierda, hacia derecha, acercar y alejar zoom y mantener (como un clic), que son los movimientos básicos para navegar por una página web.

Su uso es intuitivo y natural, convirtiendo los movimientos de usuario en acciones. Al navegar por la aplicación, los movimientos naturales de desplazar hacia la o derecha son similares a las flechas de avance y retroceso de cualquier navegador web. Para realizar una selección, el usuario sólo tiene que mantener su mano sobre el objeto, y para obtener información de productos, el usuario puede utilizar las opciones de zoom moviendo su mano más cerca o más lejos de la interfaz.

Ilustración 17 WinSense

Comunicación: Traducción de lenguaje de signos

Un grupo de investigadores franceses ha diseñado una aplicación sobre Kinect que permite traducir lenguaje de signos interpretado por un usuario a texto plano. Usando librerías de código abierto se capturan los movimientos del usuario. A continuación, el gesto se identifica usando una *Fast Artificial Neural Network* (*FANN*).

Aunque se encuentra en fase experimental, es un claro ejemplo de las capacidades de Kinect en este sector.

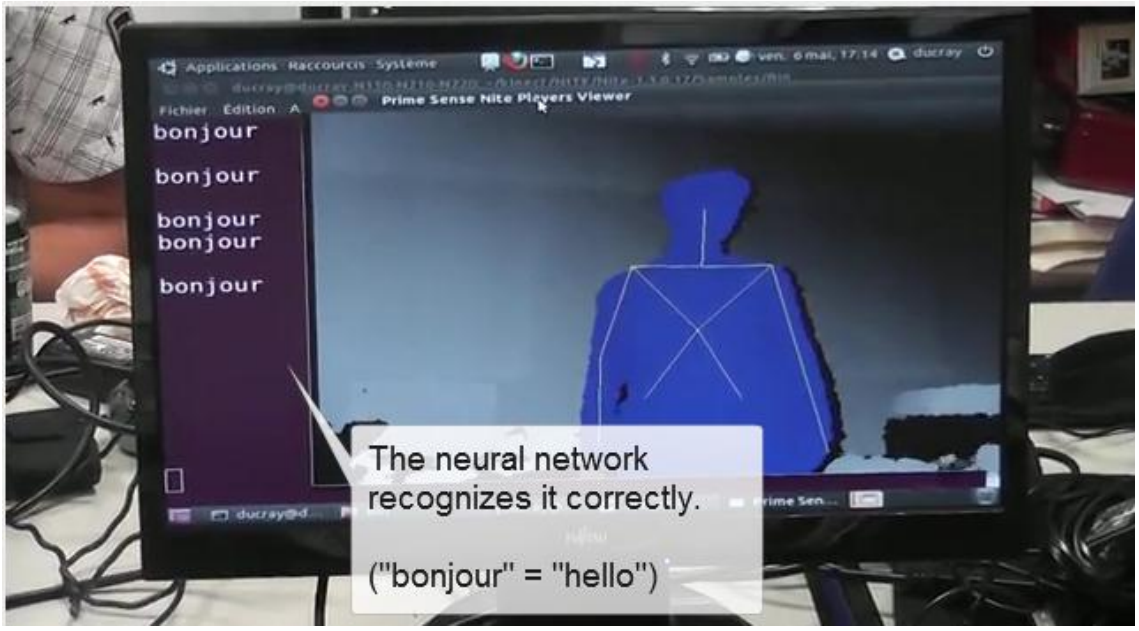


Ilustración 18 Aplicación reconociendo el signo “hola”

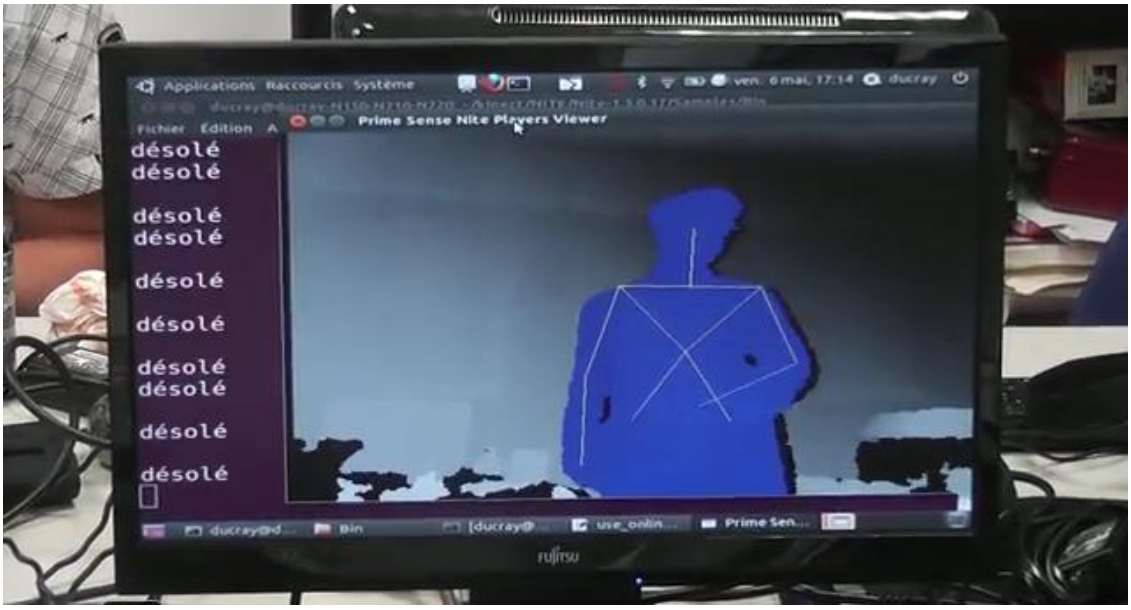


Ilustración 19 Aplicación reconociendo el signo “lo siento” (“desolé” en francés)

Accesibilidad: Ayuda de movimiento para discapacitados visuales

Otra aplicación muy interesante es NAVI (Navigational Aids fot the Visually Impaired) y tiene como objetivo usar el sensor de profundidad de Kinect para detectar paredes y obstáculos y servir como ayuda de movimiento para una persona con algún grado de discapacidad visual.

Creada por un par de estudiantes de la Universidad de Konstanz en Alemania, la aplicación hace uso, a parte del sensor Kinect, de un ordenador portátil y un dispositivo Arduino LillyPad colocados estos últimos en una mochila. El sensor se coloca en un casco especial adaptado sobre la cabeza del usuario, manteniendo de esa forma el sensor Kinect apuntando hacia el frente.

El dispositivo resultante tiene el objetivo de aumentar la capacidad de una persona ciega o con alguna discapacidad visual para navegar en entornos de interior, proporcionándoles feedback táctil sobre los obstáculos que se encuentran en su camino. Las vibraciones proceden de los motores conectados a un cinturón que están conectados a una placa Arduino.

Ilustración 20 Aplicación NAVI (1)

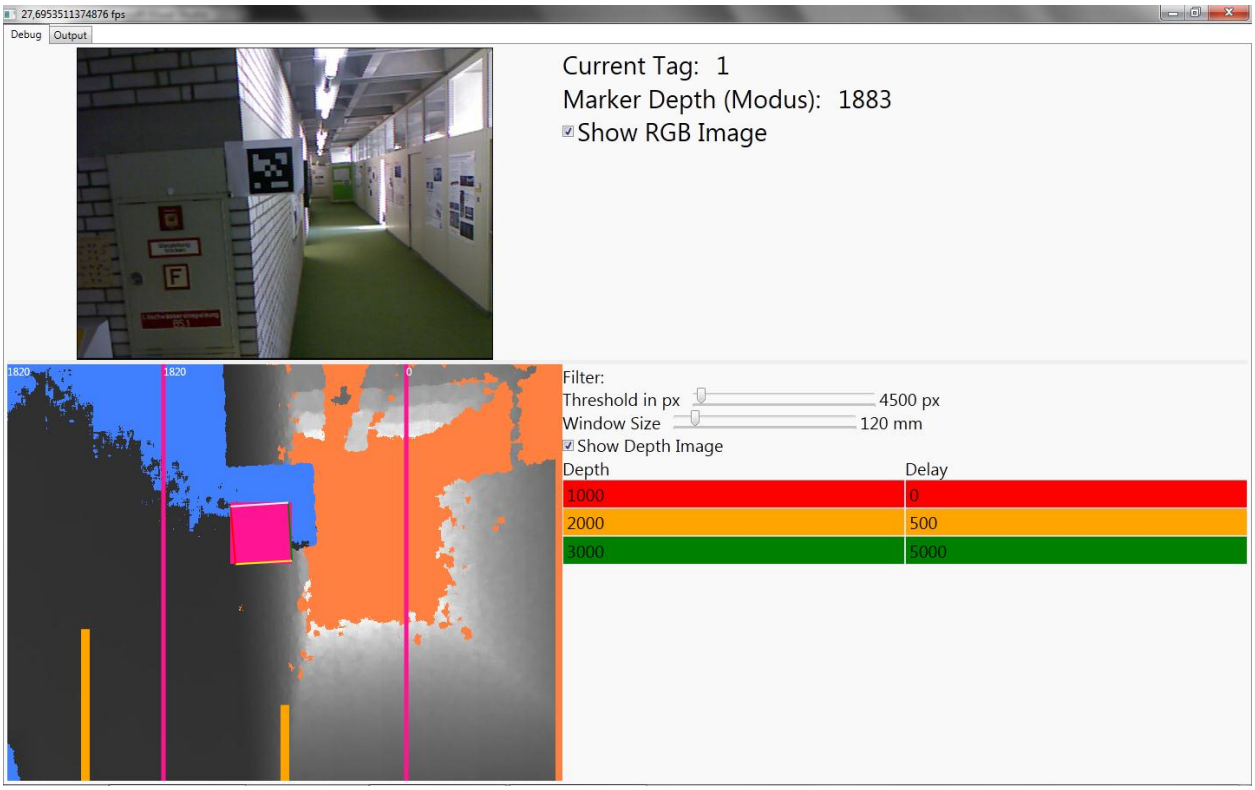


Ilustración 21 Aplicación NAVI (2)

Robótica: Control remoto de robots

Uno campo que se está beneficiando en gran medida de las capacidades de Kinect es el de la robótica. Estas aplicaciones reconocen gestos de usuario y los transmiten a robots, de forma que los movimientos del usuario son trasladados a la máquina.

Un ejemplo de control remoto de robots es Smart Pal VII, una aplicación diseñada por la compañía japonesa Yaskawa para controlar un robot humanoide usando el sensor Kinect. Los desarrolladores creen que el robot puede ser usado para comunicar y asistir al usuario en pequeñas tareas domésticas como recoger objetos del suelo.

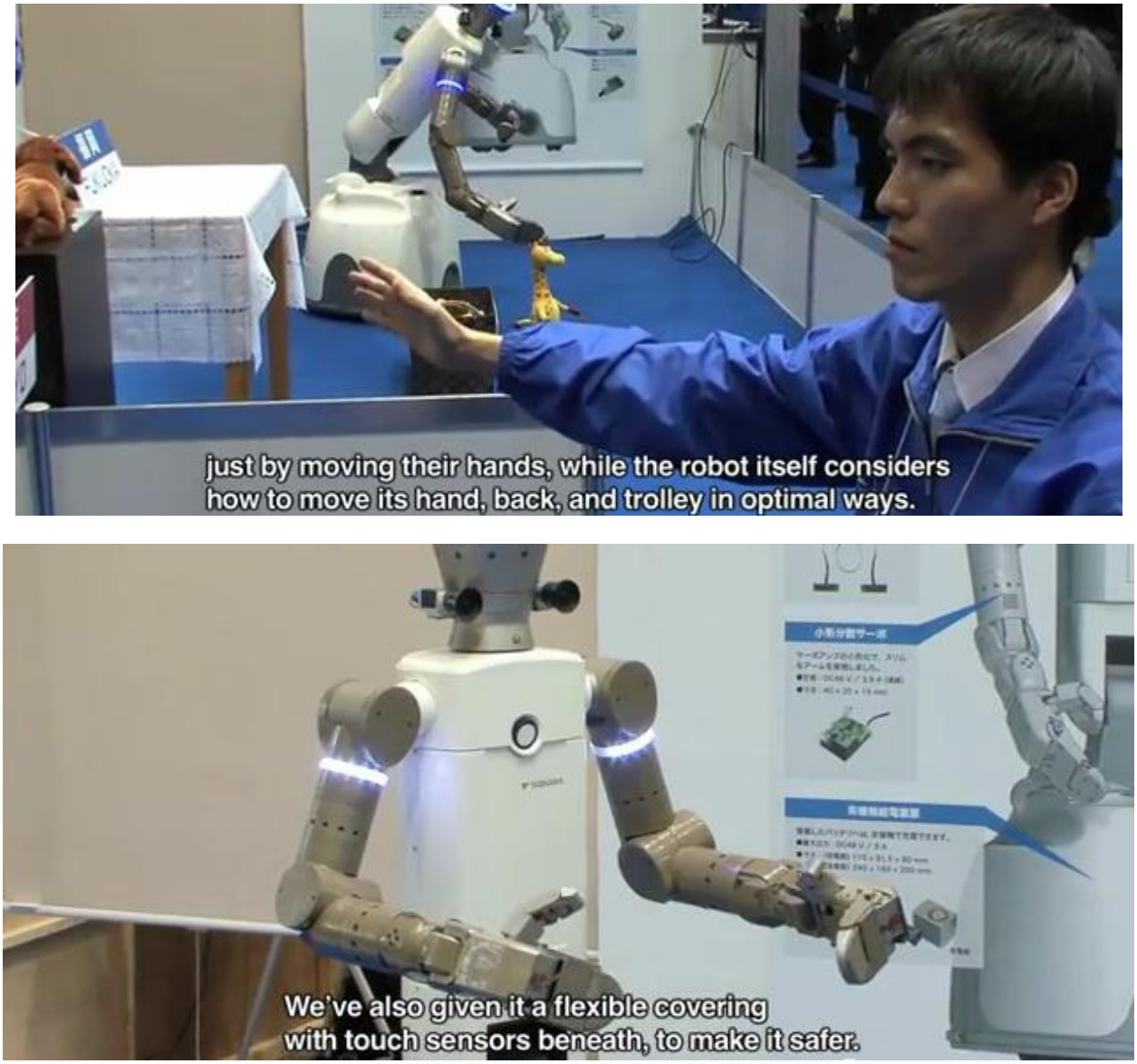


Ilustración 22 Control remoto de robot Smart Pal VII

Robótica: SLAM

SLAM, del inglés Simultaneous Localization And Mapping, o en español, Localización Y Mapeado Simultáneos. Es una técnica usada por robots y vehículos autónomos para construir un mapa de un entorno desconocido en el que se encuentra, a la vez que estima su trayectoria al desplazarse dentro de este entorno.

El problema de SLAM investiga los problemas que plantea la construcción de modelos matemáticos, geométricos o lógicos de entornos físicos, empleando como herramienta un robot móvil (en ocasiones varios de ellos) y el conjunto de sensores y actuadores que lo conforman. Dicho de otra manera, el SLAM busca resolver los problemas que plantea el colocar un robot móvil en un entorno y posición desconocidos, y que él mismo sea capaz de construir incrementalmente un mapa consistente del entorno al tiempo que utiliza dicho mapa para determinar su propia localización.

Dado que una cámara estereográfica para proyectos de investigación puede costar sobre unos 3.000 € y Kinect cuesta unos 200 €, Kinect es mucho más barata y consigue hacer lo mismo. Lo lógico era que empezaran a surgir todo tipo de proyectos de investigación que usaran Kinect en proyectos con robots.

Un ejemplo de integración de Kinect en proyectos de SLAM es el realizado por investigadores de la Universidad de Waterloo. El prototipo de llama iC2020 y es capaz de fabricar por sí solo un mapa del entorno que le rodea para así poder desplazarse por él.

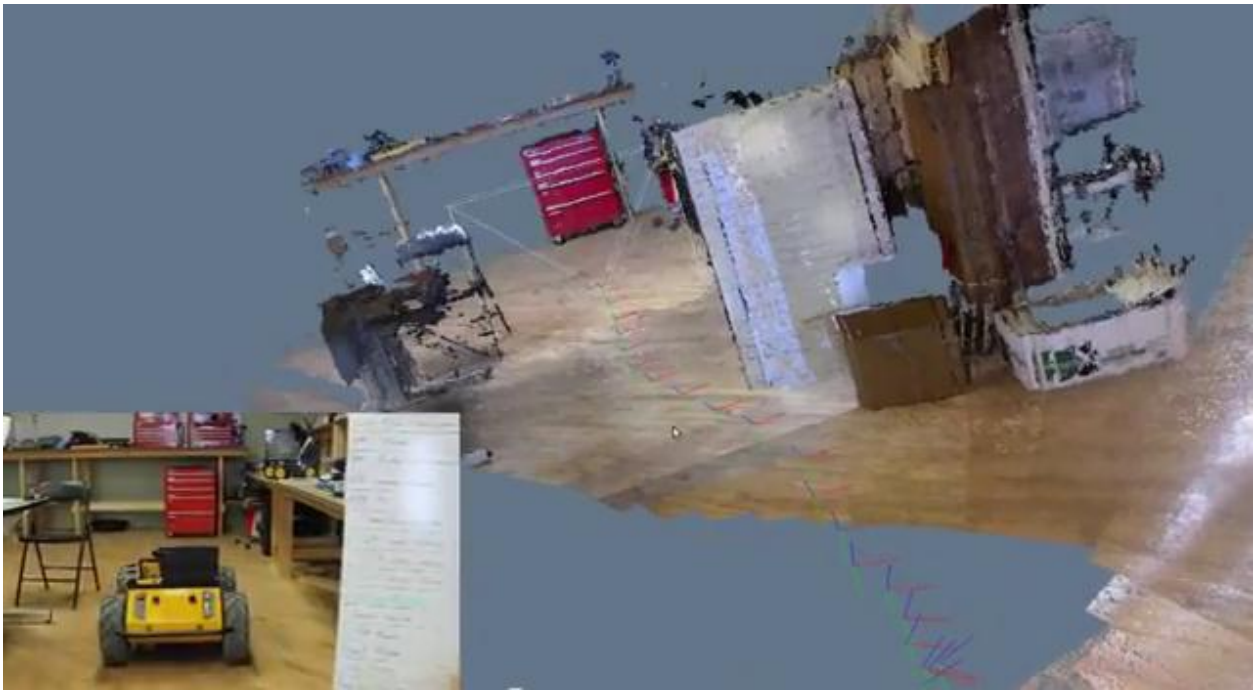


Ilustración 23 iC2020 reconociendo un escenario desconocido.

En la ilustración se muestra abajo la imagen capturada por Kinect; arriba, la representación del entorno acumulada.

Modelado 3D

La reconstrucción de objetos y entornos en tres dimensiones es uno de los campos que más se están investigando en la actualidad usando el sensor de profundidad de Kinect. Un ejemplo es *Kinect Turntable 3D Scanner*, una aplicación que partiendo de un objeto en rotación, es capaz de generar una imagen tridimensional virtual que representa al objeto original.

Ilustración 24 Objeto real (arriba) y reconstruido (abajo)

Existen además programas más elaborados que son capaces de generar una representación 3D de espacios cerrados y que son usados junto con robots móviles para, una vez obtenido un mapa del escenario, encontrar la salida del mismo.

Comercio interactivo

Otras aplicaciones están más enfocadas al comercio, como es el caso de *Fitnect*, un programa que sirve de vestuario interactivo:

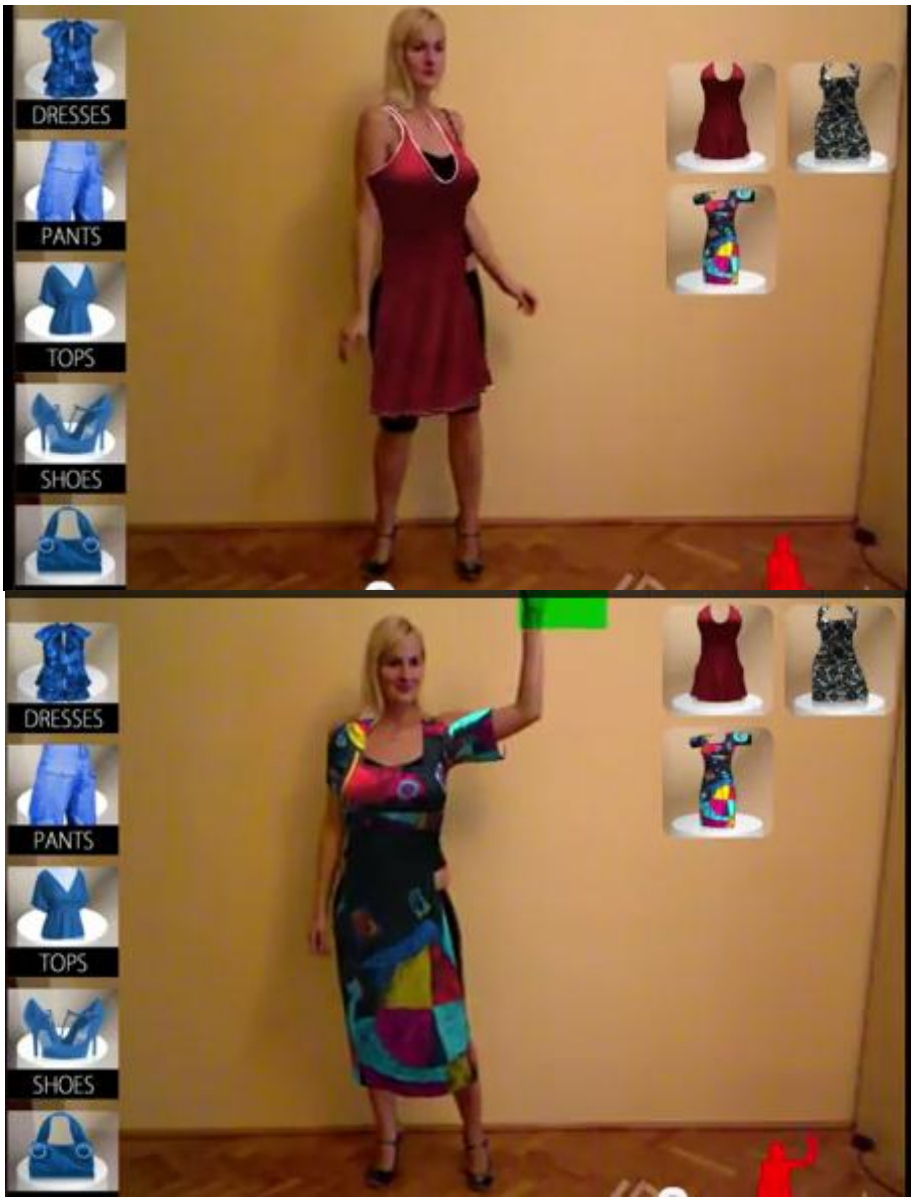


Ilustración 25 Fitnect

El funcionamiento es sencillo: el usuario se coloca en frente del sensor Kinect de forma que pueda visualizar la pantalla en la que se le muestra la interfaz. El usuario selecciona con sus manos los tipos de prendas, que son automáticamente colocados sobre la figura del usuario, mostrando de esa forma cómo quedaría la prenda si el usuario se vistieses con ella, pero sin tener que hacerlo realmente.

Interfaces de usuario sencillas y atractivas como *Fitnect* son un claro ejemplo de la viabilidad comercial de Kinect.

Medicina

Siguiendo con el uso de Kinect como interfaz de usuario natural, el área de sanidad y medicina también encuentra escenarios en los que el relativo bajo precio de Kinect es adecuado para desarrollar aplicaciones que a través de gestos de usuario realicen tareas específicas, desde tratamiento de radiografías hasta operaciones de cirugía.

Como ejemplo aquí mostramos *Virtopsy*, un programa que tiene como objetivo servir de soporte en una sala de operaciones de un hospital. Una dificultad que se encuentran los cirujanos cuando están realizando una operación, es que necesitan tener en todo momento monitorizado el área de operación, y pare ello necesitan controlar unos aparatos de visualización de forma mecánica.

Virtopsy facilita este procedimiento al cirujano permitiendo que éste pueda desplazarse por una imagen o un monitor usando gestos de sus manos.

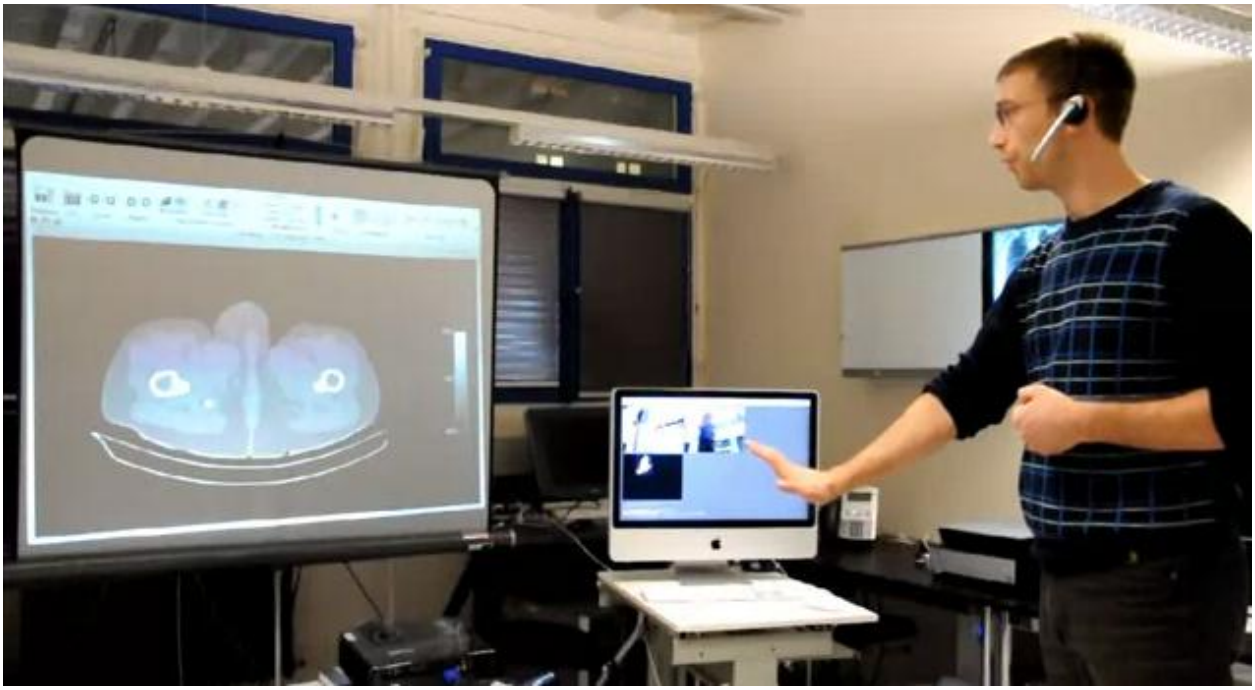


Ilustración 26 Virtopsy

Desde hace algún tiempo se viene utilizando Kinect en el ámbito de la salud. Esta vez una investigación realizada por Benjamin Blundell de la Universidad de Manchester reveló que el dispositivo desarrollado por Microsoft puede ser utilizado para curar el síndrome del miembro fantasma.

El síndrome del miembro fantasma que generalmente es tratado con la terapia del espejo, es la percepción de sensaciones de que un miembro amputado todavía está conectado al cuerpo y está funcionando con el resto de este, entre el 50 y 80% de las personas amputadas experimentan estas sensaciones fantasmas en su miembro amputado, y la mayoría de estas personas dice que las sensaciones son dolorosas.

El proyecto de Blundell utiliza Kinect y gafas de realidad virtual para ayudar a los pacientes a obtener un alivio necesario. El paciente es colocado en un entorno virtual y con el dispositivo de juegos y una serie de giroscopios se consigue “re-entrenar” al cerebro para percibir la extremidad perdida en posiciones menos dolorosas llegando incluso a tener un mínimo control sobre ella.

Una cosa que Kinect no se puede hacer es detectar la rotación, (por ejemplo detectar que alguien gira su muñeca para ver la palma de su mano), este problema puede ser resuelto con un giroscopio.

El sistema se encuentra aún en fase inicial y a pesar no haberse sometido a estrictas pruebas médicas, se ha conseguido resultados bastantes buenos con un paciente durante una prueba informal. La idea fue aceptada para que sea presentada en la GRAPP (International Conference of Computer Graphics Theory and Applications) que se llevará a cabo del 24 al 26 de febrero en Roma.

El siguiente video muestra la arquitectura, así como el funcionamiento de esta novedosa idea, que en el caso de que todo vaya bien se convertiría en una aplicación real para los dueños de Kinect.

Ilustración 27 Phantom Limb v2.0

Kinect cambió las reglas de juego y no por lo que innovó a la hora de la creación de sus videojuegos sino con la tecnología que desarrolló Microsoft. Cuando la compañía lanzó este dispositivo de seguimiento y lectura de movimientos vimos como muchos hackers y expertos los empezaron a modificar para aplicar a otros aspectos que no tienen nada que ver con los juegos.

Uno de los campos en los que más se está usando es en el de la salud y ahora se está usando para realizar análisis de cómo las personas caminan y también para la rehabilitación.

Esa virtud de poder controlar aplicaciones sin tocar ninguna clase de mando ni de pantalla táctil es la que están explotando diversos desarrolladores. Los avances en relación a los análisis se están realizando por estudiantes tanto de la Universidad de Missouri y de la Oak Ridge High School.

¿Por qué se están realizando estas investigaciones si ya existen aparatos que permiten monitorear a las personas con problemas para caminar? Porque son carísimos y, además, se necesitan ambientes creados especialmente para eso.

Para realizar las pruebas lo que hicieron fue instalar diferentes cámaras de Kinect en una casa de abuelos para poder estudiarlos y analizarlos. La información que recolectan estas cámaras puede ser utilizada para prevenir problemas de salud en los residentes. El sistema funciona tan bien que ganaron una competencia de Siemens que les dio 100.000 dólares americanos para que puedan seguir desarrollando el sistema.

Por otra parte, en el Royal Berkshire Hospital en Reino Unido, Kinect está siendo utilizado para hacer que pacientes que tuvieron un infarto y están en rehabilitación se mejoren.

En España, sin ir más lejos, existen varios proyectos de ejercicios de rehabilitación con la Kinect. Uno de los más interesantes es KinectHabilitación, creado por la Facultad de Ciencias de la Salud de la Universidad Rey Juan Carlos. En este caso, Kinect se comporta como un completo rehabilitador en la esclerosis múltiple. El paciente realiza desde su casa una batería de sesiones de ejercicios por videoconferencia, siguiendo un calendario acordado con su fisioterapeuta.

Kinect también se emplea en la actualidad con enfermos de Alzheimer. La Asociación de Servicio Integral Sectorial para Ancianos completó en Madrid un ensayo clínico aplicando Kinect en una terapia para mantener activos a los pacientes. El objetivo del proyecto KND, con una duración de nueve meses, era mejorar la calidad de vida de los ancianos afectados por esa enfermedad degenerativa.

En la Comunidad Valenciana, los Hospitales NISA han llevado a cabo un proyecto para la rehabilitación motora y cognitiva de pacientes con daño cerebral adquirido utilizando Kinect. La finalidad es que los pacientes vuelven a ser independientes en las actividades básicas de la vida diaria.

Por su parte, la empresa española Tedesys pretende introducir el sensor Kinect en los quirófanos. Su aplicación TedCas emplea Kinect en aplicaciones de cirugía asistida por ordenador. El cirujano no necesita tocar nada para controlar el sistema en tiempo real, con lo que se reducen las infecciones. A partir del kit de herramientas para desarrolladores oficial de Microsoft para Kinect, el equipo ha logrado emular el funcionamiento de una pantalla táctil. Su sistema está compuesto por una máquina de bajo consumo sin ventilador conectada a Kinect para manejar una aplicación web.

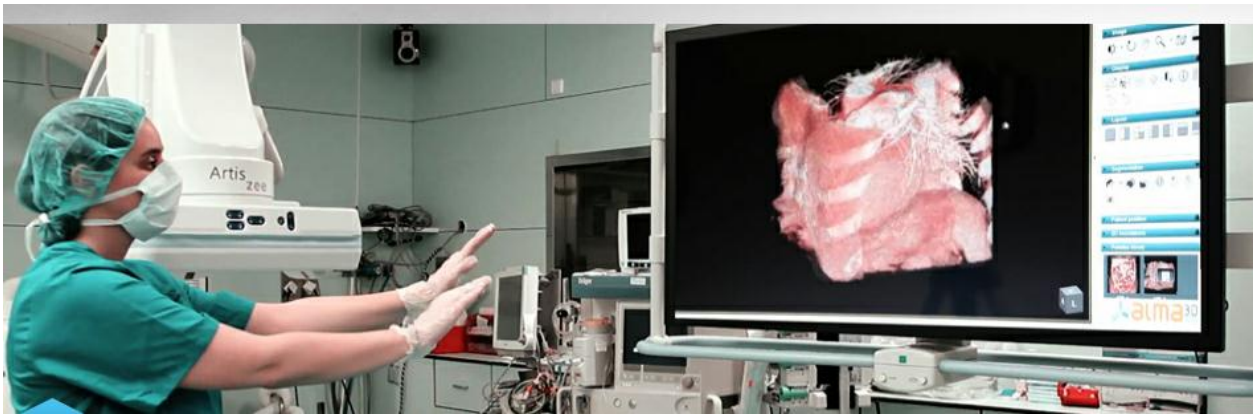


Ilustración 28 Ejemplo de NUI, TedCas

Control remoto de dispositivos

Parece que Kinect comienza también a calar en el seno del Departamento de Defensa de Estados Unidos y el Ejército tiene en mente probar este dispositivo dentro de sus helicópteros .

¿Instalar Kinect en la cabina de un helicóptero? Aunque pueda resultar extraño y a priori uno se imagine al piloto controlando el aparato con sus gestos y no con los mandos de control, la idea es que Kinect sea un complemento a los mandos de la cabina y permita al piloto realizar ciertas acciones mediante gestos, como por ejemplo abrir las puertas del helicóptero o desplegar un arma.

Teniendo en cuenta que el funcionamiento de Kinect es precisamente reconocer gestos, un sistema de este calibre ayudaría al piloto a reaccionar rápidamente y, por tanto, ofrecer un menor tiempo de respuesta. ¿Y por qué usar Kinect? La respuesta es bien simple y es algo que se repite en muchas otras investigaciones, el coste. Kinect es un sistema de visión artificial que, comparado con otras soluciones o con desarrollar un sistema desde cero, es muy barato y, por consiguiente, ofrece una gran base de partida para implementar cualquier tipo de sistema.

Las nuevas tecnologías procedentes del mundo de los videojuegos tienen un gran potencial para reducir sustancialmente los costes para añadir un sistema de reconocimiento y seguimiento de los gestos de los pilotos, abriendo la puerta a un futuro de helicópteros con cabinas inteligentes.

Si las pruebas resultan satisfactorias, el Ejército podría emplear Kinect como sistema de desarrollo de su sistema de reconocimiento de gestos y seguimiento de los movimientos de los pilotos de sus helicópteros, un proyecto bastante curioso que, al menos, apuesta por el ahorro de costes y la viabilidad económica. ¿Y si realmente se desarrolla este sistema? ¿Microsoft desarrollará un Kinect exclusivo para uso militar? Que Kinect sea un dispositivo comercial es algo que, bajo mi punto de vista, podrían perjudicar a estas pruebas desde el punto de vista de la seguridad pero, al menos, adoptando ahora este dispositivo sí que se reducen los costes y tiempos de desarrollo.

Sin duda se está realizando actualmente abundante investigación en el ámbito de control remoto de dispositivos usando la interfaz de usuario natural de Kinect.

Ilustración 29 Control remoto de helicóptero

Movimiento autónomo de dispositivos

¿Qué pasa cuando se monta un sensor Kinect en la parte superior de un *quadrocopter* (helicóptero de cuatro ruedas)? Se consigue uno de los mejores artilugios electrónicos que se hayan diseñado. Tanto, que a mucha gente le sigue pareciendo de ciencia ficción.

A los investigadores de la Universidad de Berkeley se les ocurrió la idea de diseñar un *quadrocopter* que fuera capaz de controlar de forma autónoma su movimiento. Para ello sería necesario que, mientras flota, el dispositivo pueda sentir lo que hay a su alrededor. Utilizando las funciones de Kinect, el *quadrocopter* se puede mover, cambiar el punto de vista, y reconocer un objeto o un obstáculo en su camino.

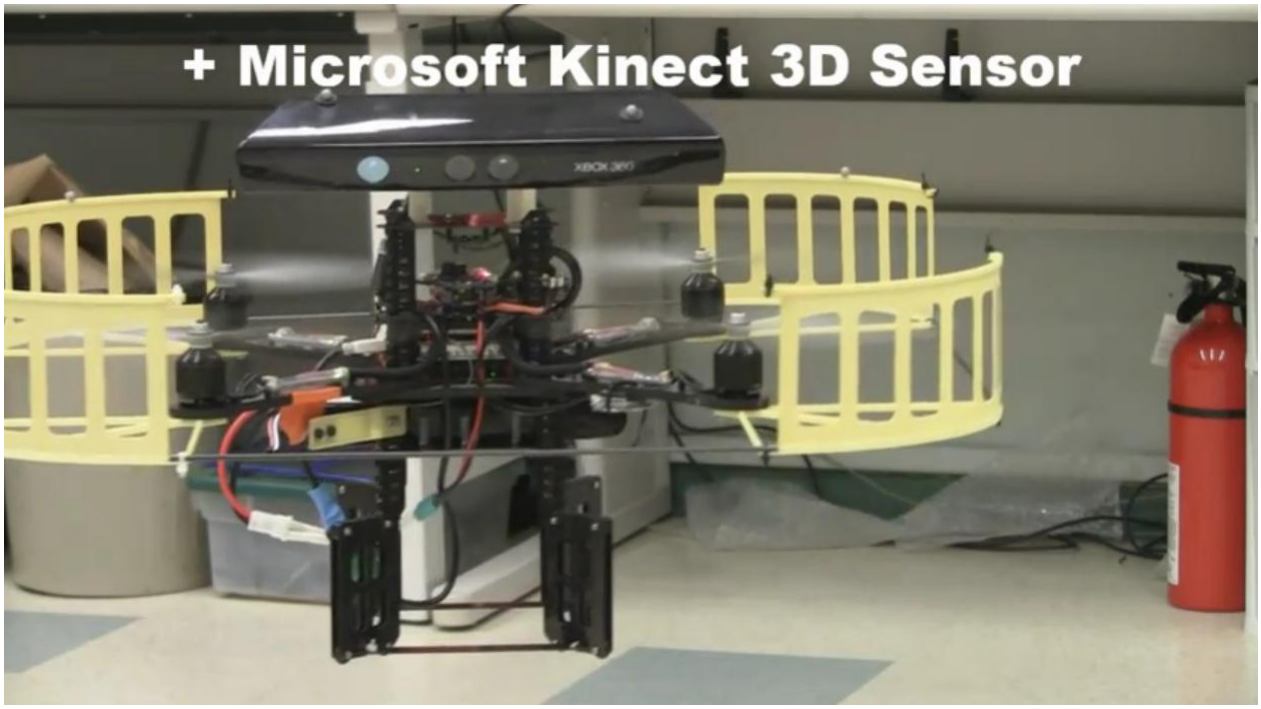


Ilustración 30 Control remoto de helicóptero

La modificación que le han realizado es usar Kinect como radar 3D. De esta forma puede detectar con precisión cómo de cerca está cada cosa. Por ahora, sólo puede ejecutar una trayectoria predeterminada. Si ve algo en su camino, se detendrá y se quedará flotando, en lugar de tratar de encontrar una forma de evitar el obstáculo. Entonces, cuando el obstáculo se mueve seguirá.

En palabras de los investigadores: “Es una pasada, pero si fuera aún más sorprendente, tendríamos que empezar a preocuparnos por el apocalipsis causado por robots.”

3. Capítulo 3: Kinect for Windows SDK

El SDK de Kinect para Windows es el conjunto de bibliotecas y herramientas que permite programar aplicaciones en distintas plataformas de desarrollo de Microsoft utilizando el sensor de Kinect como medio de entrada. Con él, podemos crear aplicaciones WPF, Windows Forms, aplicaciones de XNA e incluso aplicaciones basadas en navegador que se ejecuten en el sistema operativo Windows.

Por extraño que parezca, no podemos crear juegos de Xbox con el Kinect para Windows SDK. Los desarrolladores pueden utilizar el SDK con el sensor Kinect para Xbox. Para poder utilizar la capacidad de Modo Cercano de Kinect requeriremos el uso del hardware específico de Kinect para Windows (en vez de la versión para Xbox 360). Además, el sensor de Kinect para Windows es necesario para implementaciones comerciales, siendo sólo posible la distribución sin ánimo de lucro en el caso de las aplicaciones desarrolladas con Kinect para Xbox 360.

En este apartado se van a describir las APIs incluidas en el SDK de Kinect para Windows. Se realizará una introducción a las diferentes versiones que ha habido del SDK mostrando las mejoras que ofrecía cada uno respecto a la versión anterior y se mostrarán algunos ejemplos.

En el apartado *Kinect NUI,* se realiza una descripción de las librerías incluidas en el SDK para obtener datos en tiempo real de los sensores instalados en el sensor Kinect. Estas librerías incluyen las de mayor relevancia para los objetivos de este PFC (el API de audio), a las que se dedica el apartado *Kinect Audio API.*

3.1.Hardware de Kinect

El SDK de Kinect para Windows aprovecha y depende de los componentes incluidos en todas las versiones previstas del dispositivo Kinect. A fin de comprender las capacidades del SDK, es importante entender primero el hardware en el que se basa. La cubierta negra de Kinect incluye un cabezal y una base, tal como se muestra en la siguiente ilustración. La cabeza mide 30,5 x 6,35 x 3.8 centímetros. La unión entre la base y la cabeza está motorizada. La cubierta esconde un emisor de infrarrojos, dos cámaras, cuatro micrófonos y un ventilador.

Ilustración 31 Carcasa del sensor Kinect

No se recomienda extraer la cubierta de Kinect. Con el fin de mostrar los componentes internos se muestra la Ilustración 33. Viendo Kinect de frente, de izquierda a derecha, se encuentran los sensores y la fuente de luz que se utilizan para capturar los datos RGB y de profundidad. El componente más a la izquierda es la fuente de luz infrarroja. Junto a éste se encuentra el indicador LED. A continuación se encuentra la cámara de color utilizada para recoger los datos RGB y, por último, a la derecha (hacia el centro de la cabeza de Kinect), se encuentra la cámara infrarroja utilizada para captar los datos de profundidad. La cámara de color es compatible con una resolución máxima de 1280 x 960 píxeles, mientras que la cámara de profundidad es compatible con una resolución máxima de 640 x 480 píxeles.

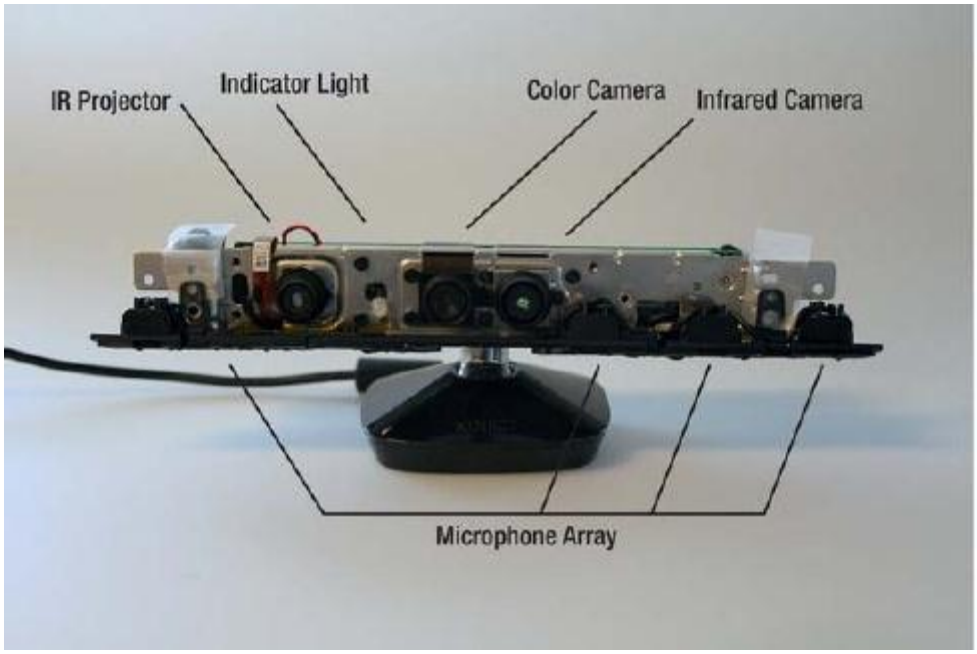


Ilustración 32 Componentes del sensor Kinect

En la parte inferior de Kinect se encuentra la matriz o conjunto de micrófonos. El conjunto de micrófonos se compone de cuatro micrófonos diferentes, como se muestra en la Ilustración 34. Uno está situado a la izquierda de la fuente de luz infrarroja y los otros tres se sitúan uniformemente a la derecha de la cámara.

El sensor Kinect que se vende independiente del paquete de Xbox, viene con un cable en forma de Y que se divide en el conector USB de Kinect y en una fuente de energía adicional para Kinect. La extensión de USB es necesaria porque el conector macho que sale de Kinect no sigue el estándar de conectores USB. La energía adicional es necesaria para ejecutar los motores en el Kinect.

Por otro lado, el sensor Kinect incluido en el paquete de Xbox es probable que no incluya un cable en Y. Esto es debido a que las nuevas consolas Xbox tienen un conector USB hembra propietario que funciona con Kinect y no requiere energía adicional para activar los servos de Kinect. Este es un problema (y una fuente de enorme confusión) si se va a utilizar Kinect para el desarrollo de aplicaciones de PC con el SDK de Kinect. Para ello será necesario adquirir por separado el cable en Y si no se incluye con Kinect. Normalmente se comercializa con el nombre *Adaptador de corriente alterna para Kinect* o *Fuente de alimentación para Kinect*. El software construido usando el SDK de Kinect no funcionará sin él.

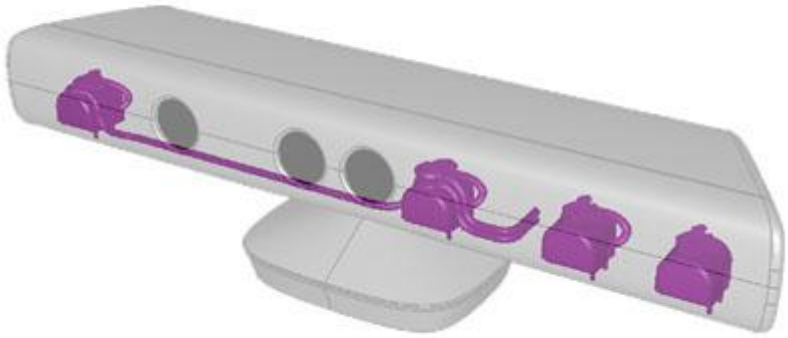


Ilustración 33 Conjunto de micrófonos de Kinect

Una última pieza de hardware interesante para Kinect vendida por la compañía Nyco y no por Microsoft se llama *Kinect Zoom*. El hardware base de Kinect realiza reconocimiento de la profundidad entre 0,8 y 4 metros. *Kinect Zoom* es un conjunto de lentes que se colocan sobre Kinect, permitiendo que el sensor Kinect sea utilizado en salas más pequeñas que las que se recomienda Microsoft. Es de especial interés para usuarios del SDK de Kinect que quieran utilizarlo para funciones especializadas, como las implementaciones personalizadas de lógica de seguimiento de dedos, o como herramienta de productividad que involucran a una persona sentada en frente de Kinect.

3.2. Versiones de Kinect for Windows SDK

Este apartado describe brevemente la evolución del SDK para Windows desarrollado por parte de Microsoft para desarrollar aplicaciones sobre Kinect, desde su primera versión beta hasta la última versión anunciada hasta la fecha.

3.2.1. Kinect for Windows SDK Beta

Microsoft anunció la primera versión del SDK para Kinect for Windows en primavera de 2011. Su objetivo era animar a desarrolladores a usar las librerías propias de Microsoft para desarrollar aplicaciones sobre Kinect y de esa forma experimentar con esta interfaz de usuario natural. Hasta la fecha, ésta tarea sólo era posible mediante librerías de código abierto como OpenNI o NITE.

Ilustración 34 Kinect for Windows SDK beta

Ilustración 35 Software open source para Kinect OpenNI

El SDK está diseñado para permitir a la comunidad de desarrolladores, investigadores académicos y entusiastas crear aplicaciones competentes que puedan incluir detección de profundidad, seguimiento de movimiento humano y reconocimiento de voz utilizando la tecnología de Kinect en Windows. Esta primera versión del SDK era estrictamente no comercial, enfocado al desarrollo de aplicaciones no comerciales, e incluía controladores, API completa para obtener flujos de datos del sensor, documentación y materiales de instalación.

Desde el principio Microsoft pensó que el SDK se usara mediante Microsoft Visual Studio 2010. El API permitía al usuario utilizar los lenguajes C++, C# o Visual Basic.

Ilustración 36 Kinect for Windows SDK beta

Se anunció la siguiente versión del SDK en Noviembre de 2011, coincidiendo con el aniversario del lanzamiento de Kinect para Xbox 360. Esta versión sería denominada *Microsoft for Windows SDK Beta 2*, sería también no comercial y simplificaba algunos aspectos del uso de las APIs, mejorando el rendimiento general de la aplicación y abstrayendo al desarrollador de algunos aspectos de inicialización de hardware.

3.2.2. Kinect for Windows SDK 1.0

En febrero de 2012 Microsoft anuncia una nueva versión del sensor, *Kinect for Windows,* y *Kinect for Windows SDK 1.0*, la nueva y hasta la fecha última versión de librerías y herramientas para desarrolladores con las que diseñar aplicaciones para Kinect desarrolladas en los lenguajes C++, C# y Visual Basic usando la herramienta Microsoft Visual Studio 2010. Esta nueva versión del SDK puede ser usado tanto con el sensor para Xbox 360 como para la versión de Windows, mientras que los anteriores SDK sólo pueden usarse con la versión de Xbox 360.

Ilustración 37 Sensor Kinect para Windows (2012)

Respecto a su anterior versión (*Kinect for Windows SDK beta 2*), esta versión incluye una serie de mejoras y nuevas funcionalidades que se detallarán en esta sección.

En los tres meses desde que se lanzó la versión *Beta 2*, se han realizado mejoras al SDK y al *entorno de ejecución* que se han incluido en la versión 1.0:

- Soporte para hasta cuatro sensores Kinect conectado en el mismo equipo.
- Seguimiento de esqueleto significativamente mejorado, incluyendo la capacidad para que los desarrolladores controlar qué usuario está siendo seguido por el sensor.
- Modo Cercano (*Near Mode*) para el nuevo Kinect para Windows, que permite a la cámara de profundidad operar desde una distancia de 40 centímetros enfrente del dispositivo.
- Actualizaciones y mejoras de la APIs (administrada y no administrada).
- Los últimos componentes de Microsoft Speech (V11) para el reconocimiento de voz se han incluido como parte de la instalación del SDK y del entorno de ejecución.
- Mejorada la precisión del reconocedor de voz.
- Nuevos y actualizados ejemplos, como *Kinect Explorer*, que permite a los desarrolladores explorar todas las capacidades del sensor y del SDK, incluyendo la funcionalidad de localización de fuente de audio y ángulos de origen, resoluciones disponibles para la cámara RGB, para la de profundidad, el seguimiento del esqueleto y los controles del motor. Todo en un mismo ejemplo con controles sencillos e interfaz usable. Es una excelente forma de mostrar las principales funcionalidades del sensor.
- Un instalador comercial listo para que el SDK pueda ser usado en programas de usuario con una mínima configuración.
- Mejoras de robustez, como estabilidad de los controladores o correcciones en el entorno de ejecución y en audio.
- Controladores y documentación técnica para desarrollar aplicaciones que usen Kinect.
- APIs y documentación de referencia para código en modo administrado (*managed code*) y no administrado (*unmanaged code*). Estas APIs proporcionan al desarrollador:
    - Flujos de contenido audiovisual del sensor. Los desarrolladores tienen acceso a los flujos de datos en bruto del sensor de profundidad, del sensor de la cámara de color y del conjunto de cuatro micrófonos.
    - Seguimiento de formas humanas. El SDK tiene la capacidad de realizar un seguimiento de la imagen de una persona en forma de esqueleto formado por 20 puntos que se desplaza dentro del campo de vista de Kinect, haciendo posible crear aplicaciones basadas en gestos de usuario.

  - Funciones de audio avanzadas. Se incluyen capacidades de procesamiento de audio como la supresión del ruido ambiente y cancelación de eco, algoritmos para identificar la fuente de sonido actual, y la integración con la API de reconocimiento de voz de Windows.
- Ejemplos preparados y listos para ejecutar que muestran buenas prácticas a la hora de usar las principales funcionalidades del sensor Kinect. Además incluye un navegador de ejemplos que permite buscar, instalar y ejecutar dichos ejemplos.
- Manuales y guías que explican el funcionamiento de los ejemplos. El SDK incluye más de 100 páginas de documentación de alta calidad técnica. Además de los archivos de ayuda integrados, la documentación incluye tutoriales detallados para la mayoría de las muestras proporcionadas con el SDK.

Otro aspecto clave de esta nueva versión es que por primera vez se distribuye con licencia comercial. Las versiones beta anteriores tenían una licencia no comercial y estaba enfocada hacia investigadores académicos y entusiastas de desarrollo de aplicaciones creativas. No incluía soporte y no otorga ningún derecho para el uso o distribución comercial de aplicaciones. La actualización de software SDK está diseñada para permitir a las organizaciones desarrollar aplicaciones comerciales para su uso con sensores de Kinect para Windows.

Aplicaciones de ejemplo incluidas en Kinect for Windows SDK 1.0

En febrero de 2012, Microsoft anuncia la primera versión oficial de su SDK para Kinect:

> *"El SDK ha sido diseñado para proporcionar una herramienta a una comunidad cada vez mayor de desarrolladores, investigadores académicos y apasionados de la tecnología, que les permitirá crear nuevas experiencias con detección de la profundidad, captación del movimiento humano y reconocimiento de voz y objetos, utilizando la tecnología de Kinect en Windows 7."*

El SDK de Kinect para Windows instala varias aplicaciones de referencia. Estas aplicaciones ofrecen un punto de partida para trabajar con el SDK. Están escritos en una combinación de C # y C++ and y cumplen sus objetivos a veces en contra de mostrar de forma clara cómo utilizar el SDK de Kinect y presentando las mejores prácticas para la programación con el SDK. Los ejemplos que se mostrarán en este apartado usan C#, pero se recomienda examinar los ejemplos escritos en C++ aunque sólo sea para recordarnos que Kinect SDK se basa en una biblioteca de C++ que fue originalmente escrito para los desarrolladores de juegos en C++. Las clases de C# son a menudo simplemente *wrappers* o clases envoltorio para estas bibliotecas de niveles inferiores y, a veces, estas abstracciones provocan confusión ya que sólo tienen sentido si tenemos en cuenta sus funcionamiento a bajo nivel en C++.

El SDK de Kinect para Windows 1.0 se instala con la herramienta *Sample Browser*. Esta herramienta es útil ya que lista los ejemplos disponibles, adjunta una pequeña descripción de cada uno y lo más importante, permite instalarlos como proyectos de Visual Studio para su edición o ejecutarlos directamente.

Kinect Explorer

Kinect Explorer es un proyecto en formato WPF escrito en C#. Muestra el modelo de programación básica para obtener imágenes del sensor en color, de profundidad y de seguimiento de esqueleto y mostrarlas en la interfaz de usuario con un criterio parecido al del concurso original de AdaFruit. La Ilustración C muestra la interfaz de usuario de la aplicación. Las secuencias de vídeo y de profundidad son utilizadas cada una para rellenar y actualizar un control de usuario (Image) diferente en tiempo real, mientras que los datos de esqueleto se utilizan para crear una superposición del esqueleto en estas imágenes.

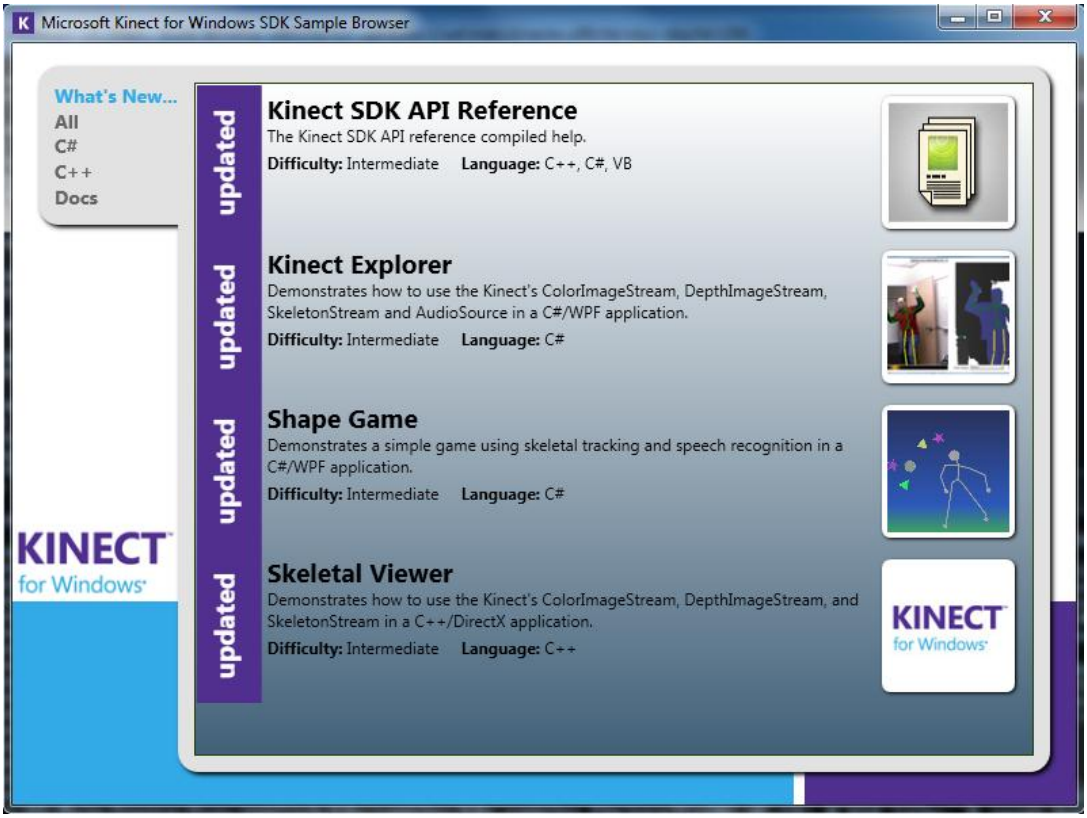


Ilustración 38 Kinect for Windows SDK Sample Browser

Aparte de los datos de profundidad, vídeo, y esqueleto, la aplicación también proporciona una actualización de los cuadros por segundo procesados por la corriente de profundidad. Mientras que la meta es de 30 FPS, tenderá a variar dependiendo de las especificaciones del sistema.

Ilustración 39 Kinect Explorer

El ejemplo expone algunos conceptos clave para trabajar con los diferentes flujos de datos del sensor. El controlador de eventos *DepthFrameReady*, por ejemplo, toma cada imagen proporcionada de forma secuencial por el flujo de profundidad y lo analiza con el fin de distinguir los píxeles del jugador de los píxeles de fondo. Cada imagen es reducida a una matriz de bytes. Cada byte se inspecciona posteriormente para determinar si se asocia con una imagen de jugador o no. Si lo hace pertenecen a un jugador, el píxel se sustituye con un color plano. Si no, es en escala de grises. La matriz de bytes se usa a continuación para crear un objeto de mapa de bits Bitmap y se define como la fuente de un control de imagen en la interfaz de usuario. Después el proceso se de inicia de nuevo para la siguiente imagen del flujo de profundidad. Lo normal es que la inspección de todos los bytes en este flujo conllevara considerable tiempo de ejecución pero, como muestra el indicador de FPS, de hecho no lo hace.

Esta la técnica predominante para manipular las corrientes de color y de profundidad. En siguientes apartados se entrará en mayor detalle sobre estos flujos de datos y cómo acceder a ellos usando el API de Kinect.

Kinect Explorer es particularmente interesante porque muestra cómo manipular las diferentes funcionalidades del sensor Kinect usando componentes reutilizables diferentes. En lugar de usar un proceso de control central para todos los flujos de datos (color, esqueleto, y audio), se permite el control independiente del propio acceso a sus respectivos flujos de datos.

Además del acceso a los flujos de datos, Kinect Explorer muestra cómo controlar el motor de movimiento de Kinect para cambiar el ángulo de inclinación de la cabeza del sensor. De esta forma el desarrollador podrá durante la programación controlar de forma manual la cabeza de Kinect.

Por último, merece un estudio cuidadoso la forma en la que los esqueletos se identifican. El SDK sólo controla esqueletos completos para dos jugadores a la vez. Por defecto, se utiliza conjunto complicado de reglas para determinar qué jugadores deben ser controlados de esta manera. Sin embargo, el SDK también permite que este conjunto predeterminado de reglas sean sobrescritos por el desarrollador de Kinect. Kinect Explorer demuestra cómo sobrescribir las reglas básicas y también proporciona varios algoritmos alternativos para determinar qué jugadores

deben recibir seguimiento esqueleto completo: por ejemplo, los jugadores más cercanos, ya que suelen ser los jugadores más físicamente activos.

Shape Game

La aplicación de referencia Shape Game, también una aplicación de WPF en C#, es un ambicioso proyecto que une seguimiento de esqueleto, reconocimiento de voz y simulación de física básica. También soporta hasta dos jugadores al mismo tiempo. El esta aplicación se combinan movimientos e instrucciones de voz para golpear objetos geométricos. El resultado del juego se muestra en la Ilustración A.

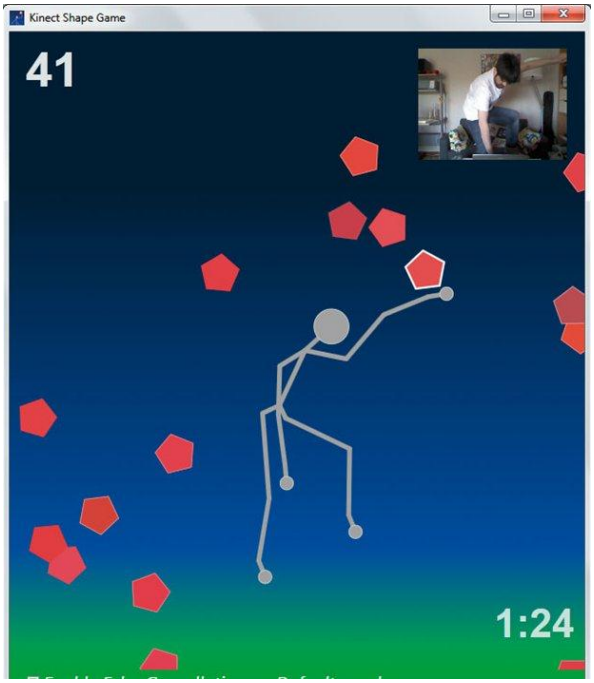


Ilustración 40 Shape Game

Este simple juego usa formas primitivas como líneas y elipses para el esqueleto. Es bastante fácil de remplazar estas formas con otras imágenes para crear una experiencia más atractiva. En el juego también se integra el reconocimiento de voz. Reconoce frases de hasta cinco palabras con una selección de aproximadamente 15 palabras posibles para cada palabra, lo que implica que se apoya en una gramática de hasta 700.000 frases. La combinación de gestos y reconocimiento de voz ofrece una forma de experimentar la jugabilidad usando este modo de juego con múltiples modos de entrada (voz y movimiento), algo que no es ampliamente utilizado en los juegos de Kinect para la Xbox, pero en torno al cual existe un considerable entusiasmo.

Skeletal Viewer

Se trata de una versión simplificada de Kinect Explorer ya que simplemente se muestra los datos del sensor de profundidad, pero, a diferencia del anterior, en éste se presentan en controles de usuario distintos la representación de los usuarios reconocidos (fuente primaria de datos del sensor de profundidad) de la representación de los jugadores como un esqueleto de hasta 20 puntos.

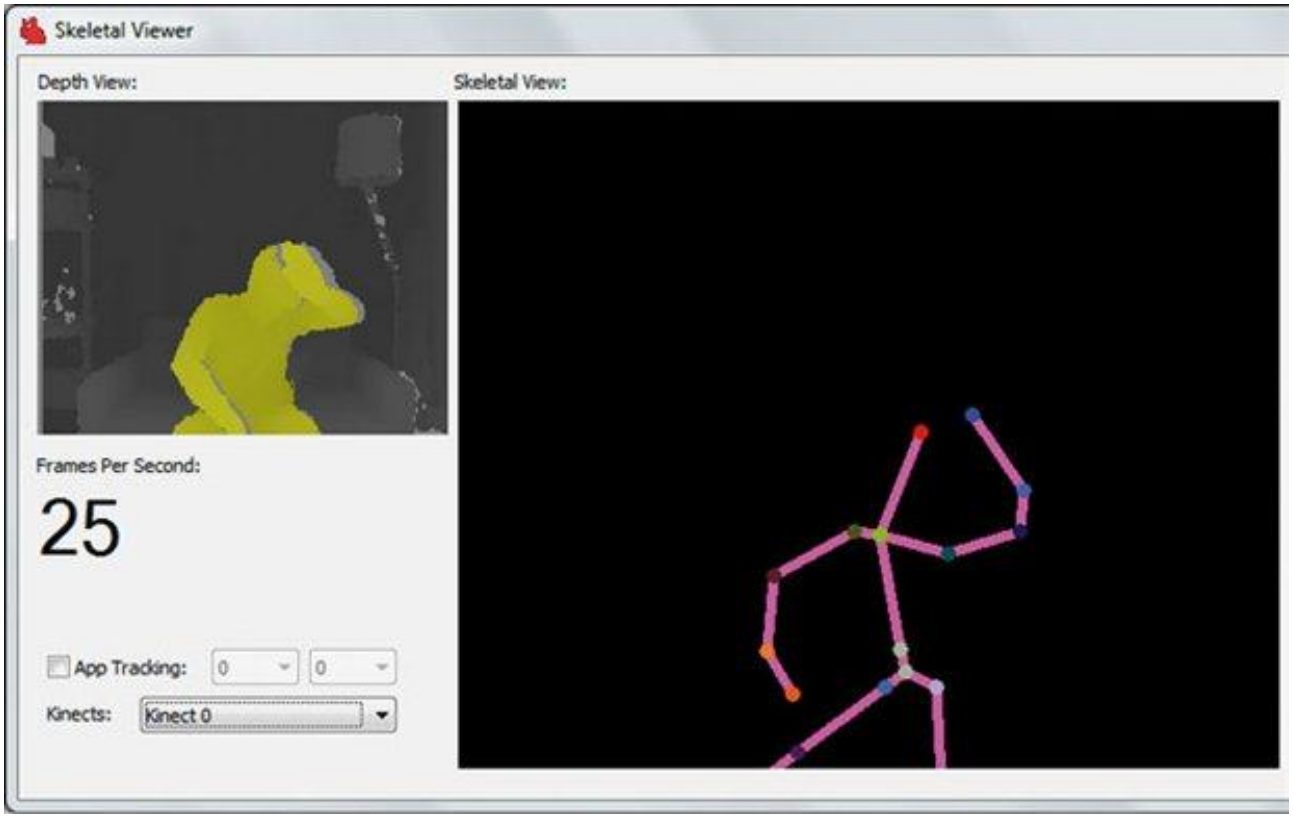


Ilustración 41 Skeletal Viewer

Audio Demo

Con este ejemplo se pone a prueba el sistema de reconocimiento de voz usando Microsoft Speech. Su funcionamiento es sencillo: Al decir “verde”, “amarillo” o “rojo”, el rótulo cambia automáticamente de color. Además, proporciona una forma muy visual de representar la localización de fuente sonora. El ángulo calculado se representa en un semicírculo y se muestra el grado de confianza generado.

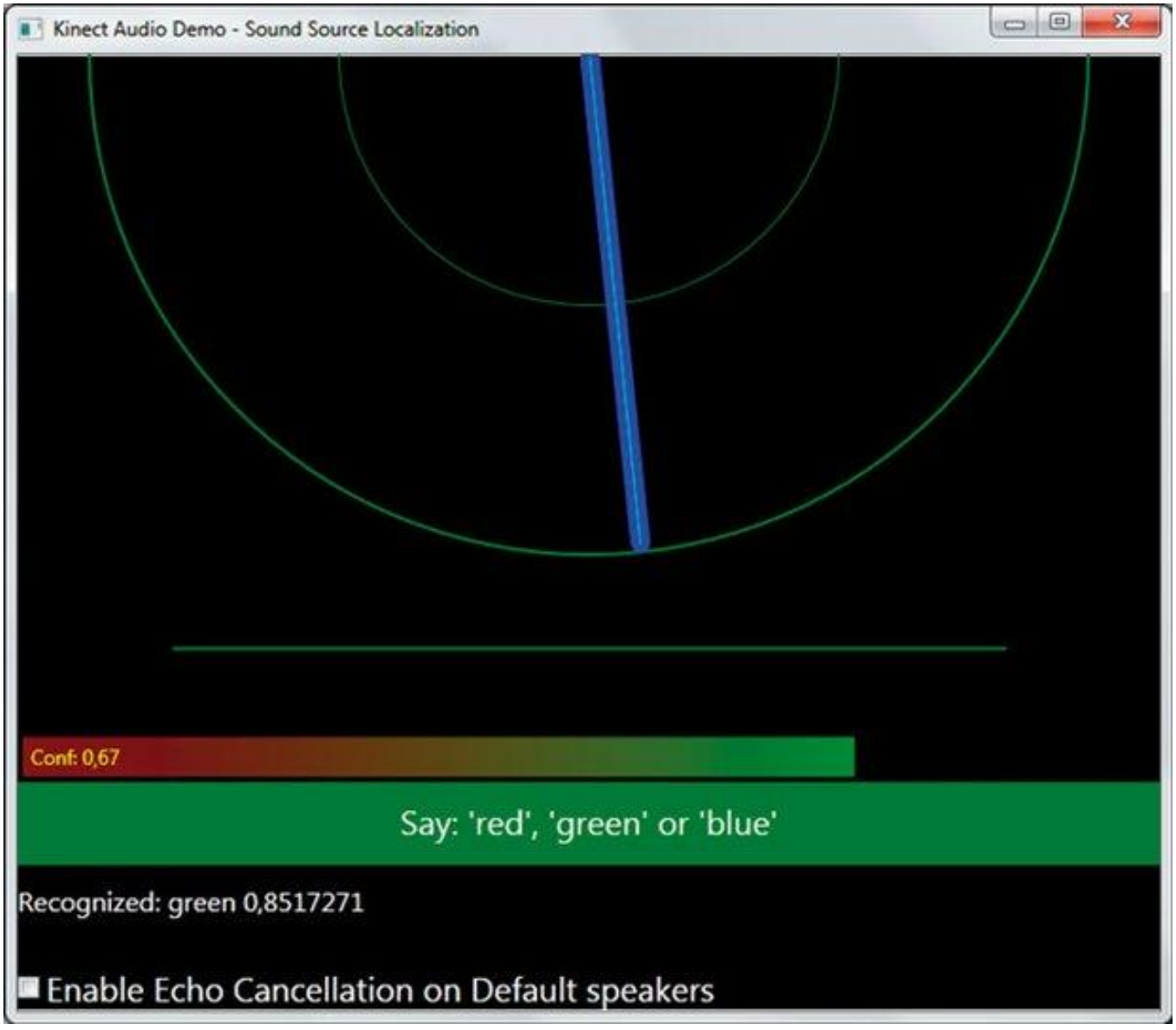


Ilustración 42 Kinect Audio Demo

Este ejemplo será el utilizado para realizar el estudio de sensibilidad del apartado 3.

3.2.3. Kinect for Windows SDK 1.5

Cuando *Kinect for Windows SDK 1.0* fue anunciado en febrero de 2012, Microsoft anunció que tenía propuesto lanzar entre dos y tres actualizaciones del SDK cada año. En palabras de Craig Eisler, General Manager de Kinect for Windows:

> *"We are also hard at work on our 1.5 release, which will be available at the end of May (2012). Among the most exciting new capabilities is Kinect Studio, an application that will allow developers to record, playback and debug clips of users engaging with their applications. Also coming is what we call "seated" or "10-joint" skeletal tracking, which provides the capability to track the head, neck and arms of either a seated or standing user. What is extra exciting to me about this functionality is that it will work in both default and near mode!*
>
> *Also included in our 1.5 release will be four new languages for speech recognition – French, Spanish, Italian, and Japanese. In addition, we will be releasing new language packs which enable speech recognition for the way a language is spoken in different regions: English/Great Britain, English/Ireland, English/Australia, English/New Zealand, English/Canada, French/France, French/Canada, Italian/Italy, Japanese/Japan, Spanish/Spain and Spanish/Mexico.".*

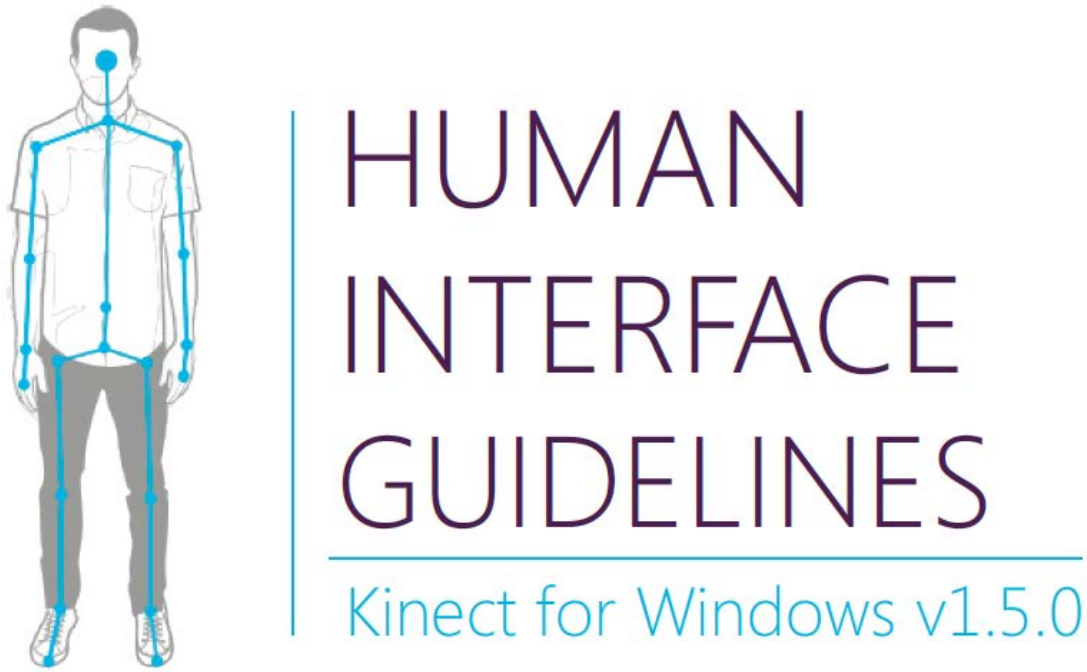


Ilustración 43 Kinect for Windows v1.5 SDK

A diferencia de otras bibliotecas de Kinect, *Kinect for Windows SDK*, como su nombre indica, sólo se ejecuta sobre los sistemas operativos Windows. En concreto, se ejecuta en las versiones x86 y x64 de Windows 7. Se ha demostrado que funciona también en las primeras versiones de Windows 8 y se prevé que venga instalado por defecto. Debido a que Kinect ha sido diseñado para el hardware de Xbox, el hardware requerido es muy similar al de un PC para poder funcionar con eficacia. No es posible usar un entorno virtualizado para ejecutar estas aplicaciones ya que los drivers de Kinect y el entorno de ejecución deben estar instalados en la máquina que ejecuta la aplicación.

Requisitos de software

- Microsoft Visual Studio 2010 Express u otra versión de 2010.
- .NET Framework 4.

Los ejemplos de reconocimiento de voz requieren:

- Software Development Kit de Microsoft Speech Platform (Version 11), si se quieren compilar los ejemplos.
- Entorno de ejecución de Microsoft Speech Platform (Version 11), que se instala con el entorno de ejecución de Kinect, si sólo se quieren ejecutar los ejemplos.

Los ejemplos *Depth-D3D* y *DepthWithColor-D3D* requieren:

- DirectX Software Development Kit, para compilación.
- DirectX End-User Entorno de ejecucións (Junio 2010), para ejecución.

El ejemplo de prueba de avatares y el ejemplo *XNABasics* requieren:

- Microsoft XNA Game Studio 4.0, para compilación.
- Microsoft XNA Framework Redistributable 4.0, para ejecución.

Requisitos de hardware

El sistema sobre el que se ejecute la aplicación debe cumplir con los siguientes requisitos mínimos:

- Procesador de 32 bits (arquitectura x86) o 64 bits (x64).
- Procesador Dual-core a 2.66 GHz o superior.
- Entrada USB 2.0 dedicada a Kinect
- 2 GB de RAM.
- Tarjeta gráfica que soporte DirectX 9.0c.
- Sensor Microsoft Kinect for Windows.

Destaca como nueva funcionalidad ***Kinect Studio***, una herramienta con la que es posible grabar flujos de datos del sensor para ser redirigidos a una aplicación en ejecución que use Kinect en cualquier momento. Es posible, por tanto, almacenar un flujo de datos tanto del sensor de profundidad, de la cámara RGB o una muestra de audio.

3.3. Natural User Interface

En este apartado se describe de forma general el concepto de interfaz natural de usuario (en inglés, *Natural User Interface* o *NUI*). Ya que Kinect en sí mismo es un dispositivo para la interacción entre hombre y máquina, en este apartado se muestra una descripción general del término NUI. Este apartado aporta una visión general e histórica de los avances que se han producido en este ámbito y se considera una información clave para entender cómo funciona Kinect.

Una interfaz de usuario natural, en español, es un término usado por diseñadores y desarrolladores de interfaces hombre-máquina para referenciar a una interfaz de usuario que:

- Es eficazmente invisible al usuario, o se convierte en invisible tras sucesivas interacciones.
- Está basada en elementos naturales o referidos a la naturaleza (por ejemplo, la física, también conocida como Filosofía de la naturaleza).

Se usa la palabra natural porque la mayoría de las interfaces de usuario usan dispositivos de control artificiales cuya operación debe ser aprendida. Una NUI se basa en que un usuario sea capaz de aprender su funcionamiento fácilmente a un nivel avanzado. Estas interfaces requieren aprendizaje, pero éste es obtenido a través de un proceso que da al usuario la sensación de que está continuamente acertando en sus acciones de forma sencilla y natural. Por ello, la palabra *natural* se refiere al tipo de experiencia del usuario percibe usando la interfaz, es decir, que el proceso de usar tecnología se realiza de forma natural conforme él entiende el concepto.

Se han propuesto varias estrategias de diseño para cumplir este objetivo con distinto grado de éxito. Un ejemplo de estrategia es el uso de *reality user interface* o *RUI*, método también conocido como *reality-based interfaces (RBI)*. Un ejemplo de estrategia *RUI* es usar un computador "usable" para hacer que objetos del mundo real sean *pulsables*, es decir, que el usuario pueda pulsar en cualquier objeto cotidiano a fin de que funcione como un hipervínculo, fusionando de esa forma el ciberespacio y el mundo real.

Una estrategia de diseño de una NUI no basada en RBI es restringir la funcionalidad y la personalización de la interfaz, de forma que los usuarios tengan muy poco que aprender para utilizar un dispositivo. Una vez se ha conseguido que las capacidades básicas del usuario estén cubiertas, se pone como objetivo que su uso carezca de dificultad. Esta es en general la estrategia de diseño del sistema operativo iOS de Apple. Ya que este diseño va de la mano de una interfaz directamente accesible por el usuario, usuarios no entendidos (no desarrolladores) a menudo erróneamente atribuyen la ausencia de esfuerzo en la interacción con el dispositivo *multitouch* por medio de la interfaz, y no con el diseño del software sobre el que realmente se asienta.

Historia

En los años 1970, 1980 y 1990 Steve Mann desarrolló una serie de estrategias de implementación de interfaces de usuario utilizando la interacción natural con el mundo real como alternativa a una interfaz de línea de comandos (*Command Line Interface, CLI*) o una interfaz gráfica de usuario (*Graphical User Interface, GUI*).

Ilustración 44 Evolución de las interfaces de usuario

Mann llamó a su trabajo *Natural User Interfaces*, *Direct User Interfaces*, and *Metaphor-Free Computing.* La tecnología de Mann llamada *EyeTap* suele entrañar un ejemplo de una interfaz de usuario natural. El uso de Mann de la palabra *natural* se refiere tanto a la acción que realiza naturalmente a los usuarios humanos, como el uso de la misma naturaleza, es decir, la física (Filosofía Natural), y el entorno natural. Un buen ejemplo de un NUI en estos ámbitos es la *hydraulophone*, especialmente cuando se usa como un dispositivo de entrada, en el que tocar un elemento natural (agua) se convierte en una forma de introducción de datos. De manera más general, un tipo de instrumentos musicales llamados *physiphones*, llamado así por las palabras griegas *physika*, *physikos* (naturaleza) y *phone* (sonido), también se han propuesto como *interfaces de usuario basadas en la naturaleza*.

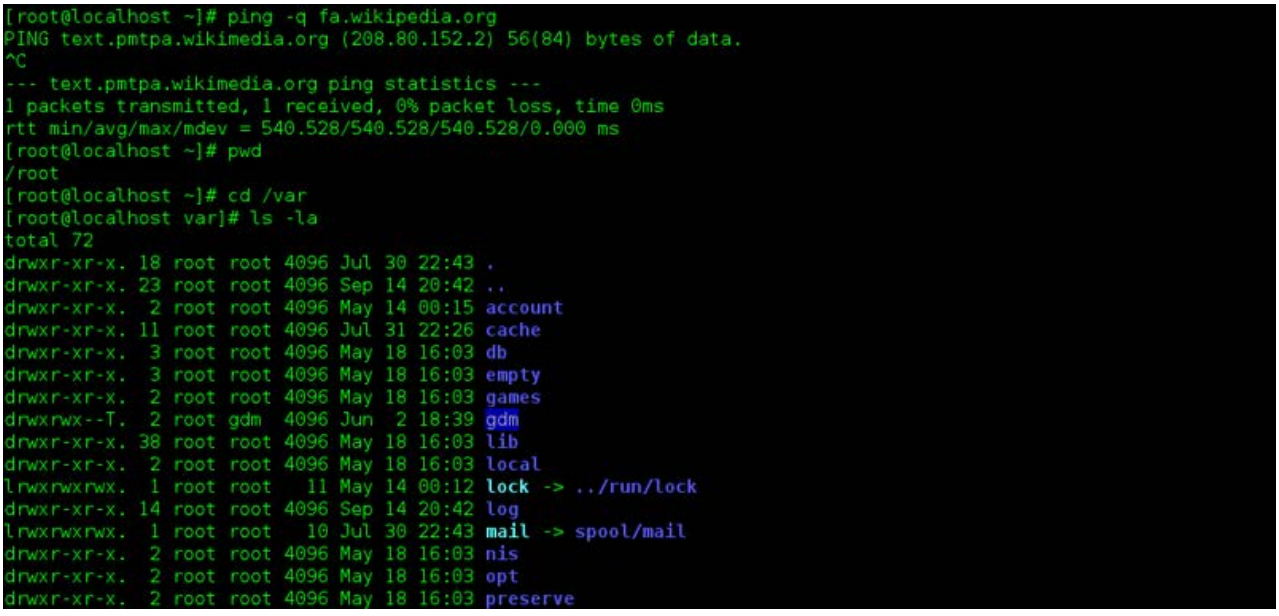


Ilustración 45 Command Line Interface de un sistema Unix (Ubuntu 10.04)

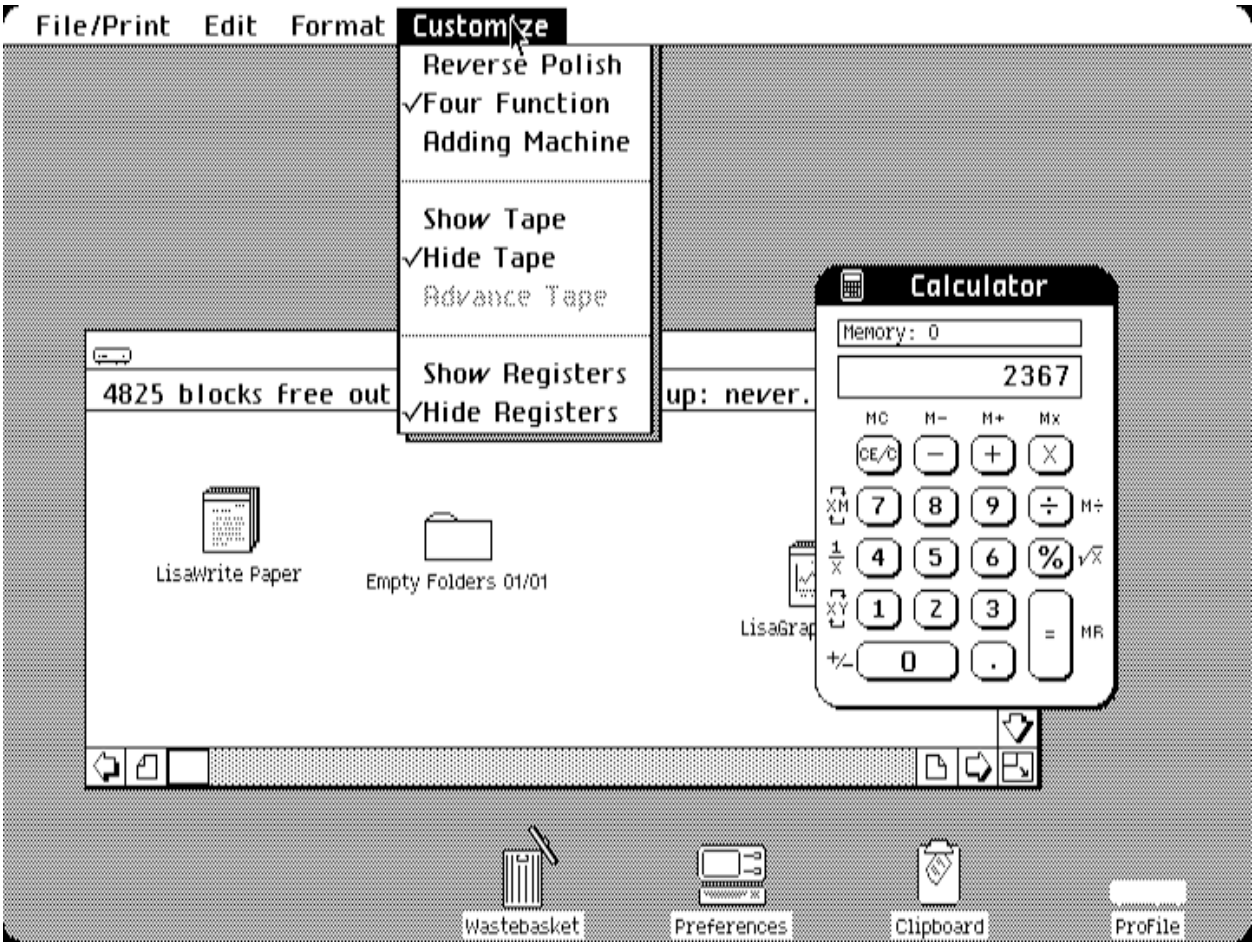


Ilustración 46 Interfaz de usuario WIMP de Apple Lisa, 1984

Entonces, cuando el ratón (mouse) permitió el surgimiento de la interfaz gráfica de usuario, los usuarios pudieron aprender fácilmente los movimientos de interacción necesarios con el ratón y otras acciones, siendo capaces de explorar la interfaz mucho más. Las GUI se basaban en metáforas para interactuar con el contenido en pantalla o los objetos. Nuevos términos como *escritorio* o la acción de *arrastrar* por ejemplo, son metáforas para una interfaz visual que eran traducidas al lenguaje codificado de la computadora.

En el año 2010 Bill Buxton, empleado de Microsoft, reiteró la importancia de las NUI dentro de Microsoft Corporation, con un video discutiendo las tecnologías que podrían utilizarse usando este tipo de interfaces, y su potencial futuro.

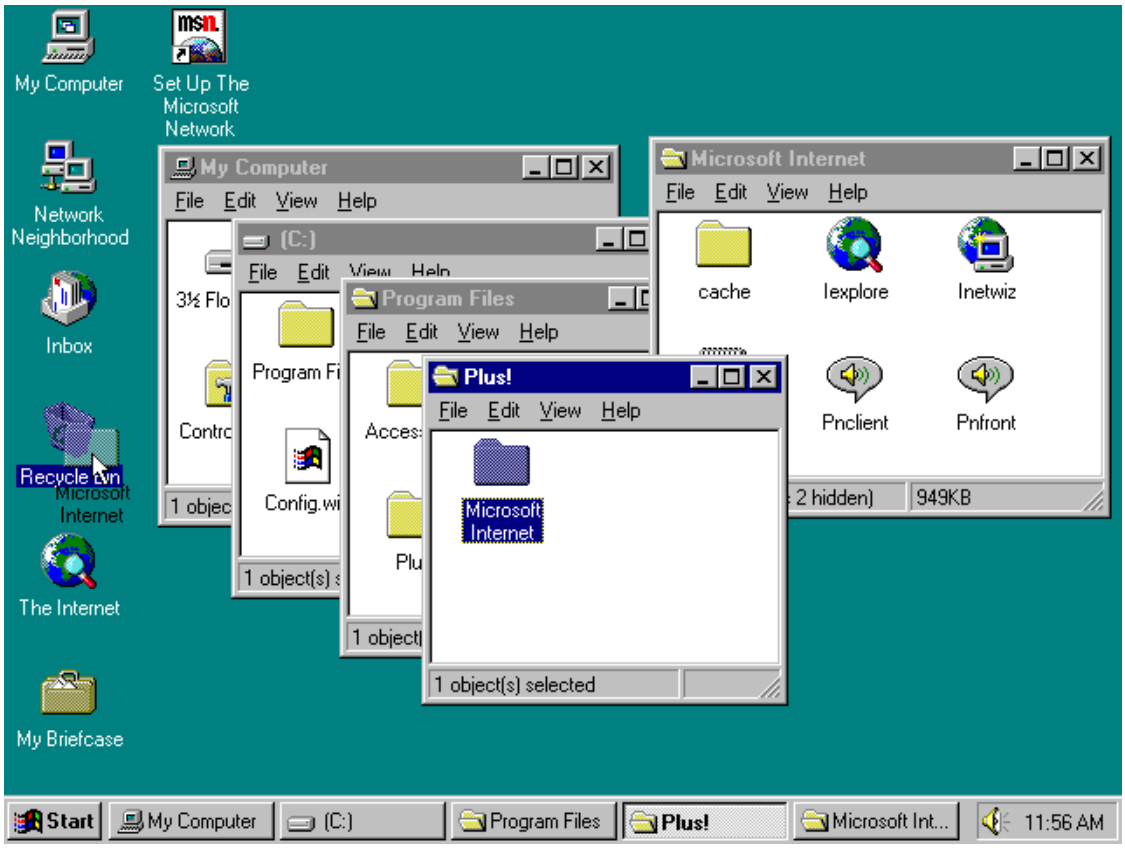


Ilustración 47 Graphical User Interface de Windows 95, 1995

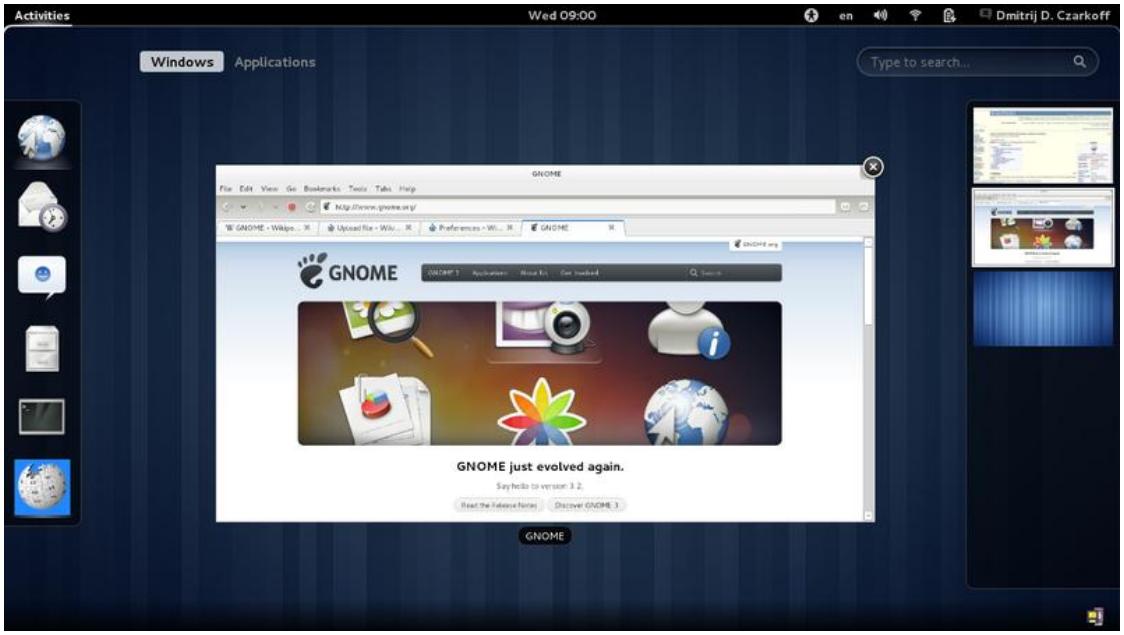


Ilustración 48 Graphical User Interface GNOME Shell, 2011

Primeros ejemplos de NUI: interfaces Multitouch

Cuando a Bill Buxton (uno de los padres de la interacción hombre-máquina) se le preguntó sobre la interfaz del primer iPhone de Apple, respondió "Las tecnologías Multitouch tienen una larga historia. Para ponerlo en contexto, la obra original realizada por mi equipo se llevó a cabo en 1984, el mismo año en que la primera computadora Macintosh salió al mercado, y no fuimos los primeros".

De hecho, la interfaz del iPhone incluye numerosos elementos WIMP, por lo que se encuentran pocos ejemplos en el dispositivo que se puedan definir realmente como parte de un NUI por diseño.

Multitouch es por tanto una tecnología que podría dar lugar a una interfaz de usuario natural. Sin embargo, la mayoría de las herramientas que se usan se utilizan para construir estas interfaces son las que tradicionalmente de usan para crear GUI.

Ilustración 49 Interfaz Multitouch de Apple iPhone, 2008

Ejemplos de interfaces reconocidas como NUI

Perceptive Pixel

Un ejemplo es el trabajo realizado por Jefferson Han en las interfaces multitouch. En una demostración en el TED (*Technology, Entertainment, Design*) de 2006, mostró varios medios de interacción con contenido mostrado en una pantalla, utilizando tanto la manipulación directa como gestos. Por ejemplo, para formar una masa glutinosa en la pantalla, Jeff directamente pellizcó y presionó con los dedos sobre la pantalla. Para hacer esto mediante la interfaz gráfica de usuario de una aplicación de diseño, un usuario tendría que utilizar una herramienta de edición usando el mouse.

Jefferson demostró que la interacción del usuario podría ser mucho más intuitiva mediante la supresión de los dispositivos de interacción a los que estamos acostumbrados y su sustitución por una pantalla que sea capaz de detectar una gama mucho más amplia de gestos y acciones humanas. Por supuesto, este sistema sólo permite que un conjunto muy limitado de las interacciones se correspondan directamente con la manipulación física directa (RBI). La ampliación de las capacidades del software más allá de las acciones físicas requiere mucho más trabajo de diseño.

Ilustración 50 Perceptive Pixel, 2006

Microsoft Surface

Microsoft Surface implementa ideas similares a Perceptive Pixel sobre cómo los usuarios interactúan con el contenido, pero añade la capacidad del dispositivo para reconocer ópticamente los objetos colocados directamente sobre la superficie de la interfaz. De esta manera, los usuarios pueden realizar acciones en el equipo usando los mismos gestos y movimientos que permitía Perceptive Pixel, añadiendo también objetos como parte de los mecanismos de control.

Así por ejemplo, cuando se coloca un vaso de vino sobre la mesa, el ordenador lo reconoce como tal y muestra el contenido asociado a esa copa de vino.

Ilustración 51 Interfaz Microsoft Surface, 2008

3D Immersive Touch

3D Immersive Touch se define como la manipulación directa de objetos 3D en un entorno virtual utilizando una o varias superficies *multitouch* en entornos virtuales en 3D con múltiples usuarios. Acuñado por primera vez en 2007 para describir y definir la interfaz de usuario natural 3D de ciertos principios de aprendizaje, Immersive Touch parece estar tomando un enfoque más amplio con la adaptación más amplia de hardware de interacción usando superficies y el tacto, como los dispositivos iPhone, iPod Touch, IPad, y una lista creciente de otros equipos.

Apple también parece estar teniendo un gran interés en las interfaces de usuario natural de 3D Immersive Touch durante los últimos años. Este trabajo se basa sobre la amplia base académica que ha estudiado la manipulación 3D en entornos de realidad virtual.

Ilustración 52 Ejemplo de interfaz basada en Immersive Touch

Microsoft Kinect para Xbox

En este apartado se describe el sensor Kinect como periférico del sistema Xbox 360 y enfatiza su uso con interfaz de usuario natural en el ámbito del entretenimiento.

Kinect es un periférico para el sistema Xbox 360 que usa gestos espaciales para realizar el proceso de interacción en lugar de un controlador de juego típico. Según Microsoft, Kinect está diseñado para ser "una nueva y revolucionaria forma de jugar: ningún controlador necesario."

Ilustración 53 Publicidad de Kinect para Xbox 360

Igual que antes, el hecho de que Kinect permita la detección del mundo físico, muestra el potencial para diseñar sistemas RBI, y por lo tanto para diseñar NUI.

Ilustración 54 Interfaz NUI Kinect de Microsoft Xbox 360

3.4. Arquitectura de Kinect for Windows

El SDK proporciona una sofisticada librería software y herramientas para ayudar a desarrolladores a utilizar las interfaces naturales de entrada de Kinect, que reconoce y reacciona ante eventos del mundo real. El sensor Kinect y la librería software interactúan con la aplicación, como se muestra a continuación:

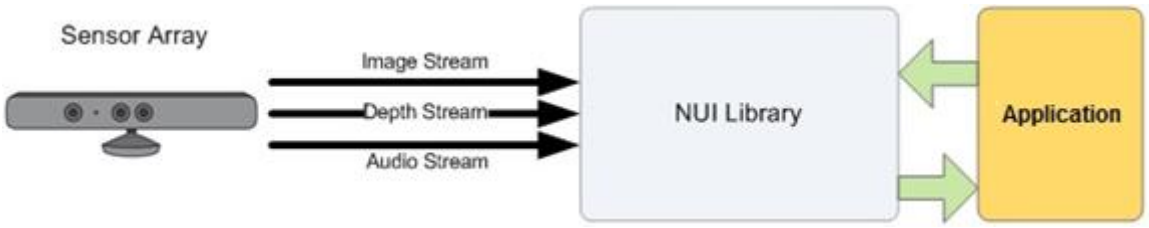


Ilustración 55 Hardware and software interaction with an application

Los componentes del SDK son:

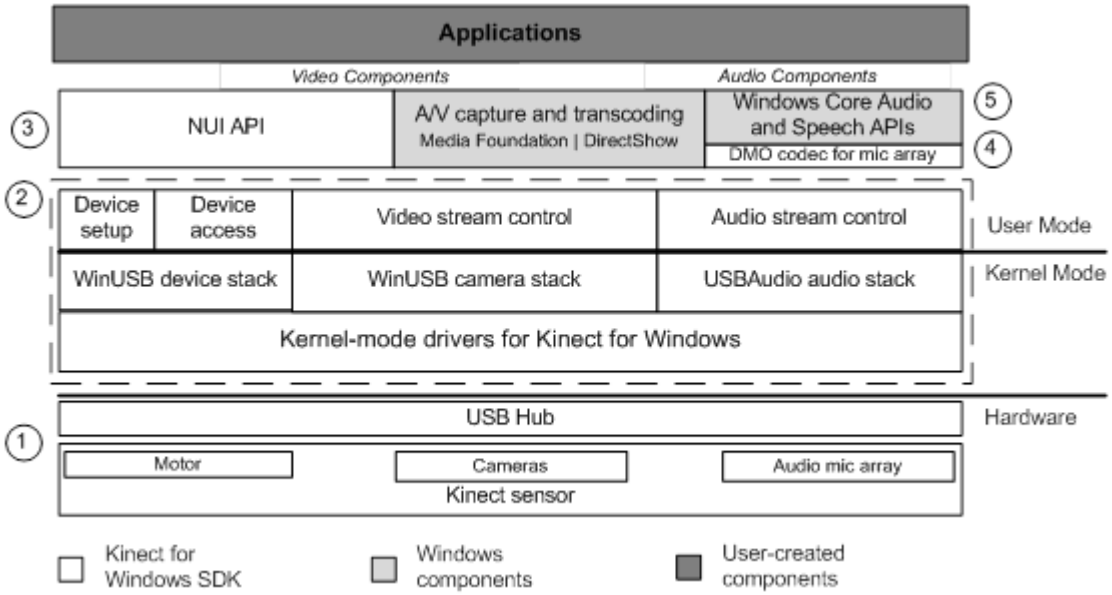


Ilustración 56 Arquitectura del SDK de Kinect

Estos componentes incluyen:

- Hardware de Kinect: Los componentes hardware incluyen el sensor Kinect y el conector USB mediante el que el sensor se conecta con el ordenador.

- Drivers de Kinect: Los drivers necesarios para realizar comunicaciones con Kinect, que son instalados en el instalador comercial del SDK. Estos controladores dan soporte a:
  - El array de cuatro micrófonos de Kinect como dispositivo de audio en modo núcleo al que se puede acceder con la API estándar de audio de Windows.
  - Streaming de imágenes y datos de profundidad.
  - Funciones de numeración de dispositivo para permitir usar más de un sensor Kinect.

- KinectAudio DirectX Media Object (DMO): El DMO de Kinect amplía el soporte al array de micrófonos para permitir las funcionalidades de *beamforming* y localización de origen de sonido.

- APIs estándar de Windows 7: Son las APIs de Windows 7 SDK y de Microsoft Speech SDK para audio, reconocimiento de voz y otro contenido audiovisual.

3.5. Kinect NUI

La interfaz de programación de aplicaciones NUI de Kinect es el núcleo del SDK para Kinect. Proporciona el soporte básico para acceder a datos de los sensores del dispositivo, incluyendo:

- Acceso a los sensores Kinect conectados al ordenador.
- Acceso a imágenes y a los flujos de datos de profundidad de Kinect.
- Entrega de una versión procesada de datos de imágenes y datos de profundidad para apoyar el seguimiento del esqueleto.

Diferencia entre código administrado y no administrado

El código gestionado es el código de un programa de ordenador que se ejecuta bajo la gestión de una máquina virtual, a diferencia del código no gestionado, que es ejecutado directamente por el sistema operativo del ordenador (CPU). Las ventajas del código gestionado incluyen facilidades para el programador y garantías de seguridad. En concreto el término código gestionado es muy dominante, aunque no exclusivo, en el mundo Microsoft. Los lenguajes más comunes de Microsoft para crear código gestionado son Java, C# y Visual Basic.NET.

En principio en cualquier lenguaje de programación los programas se pueden compilar en código gestionado o no gestionado. En la práctica, sin embargo, cada lenguaje de programación se compila en un tipo. Por ejemplo, el lenguaje de programación Java casi siempre se compila en código gestionado, aunque hay compiladores de Java que pueden generar código no gestionado (como el compilador GNU de Java).

Ejemplos de código no administrado son Visual Basic, C o C++. Usando este paradigma, el programa debe controlar por sí mismo la gestión de memoria, seguridad, control de tipos de datos, y otros servicios. Esto hace que sea más inseguro y propenso a fallos. Estos programas son ficheros ejecutables en formato de imagen binaria.

Se dice que un entorno de ejecución proporciona código administrado o gestionado si es capaz de proveer ciertos servicios automáticos al código que se ejecuta. En este caso, el entorno de ejecución es el CLR de .NET.

Los servicios son variados:

- Cargador de clases: permite cargar en memoria las clases.
- Compilador MSIL a nativo: transforma código intermedio de alto nivel independiente del hardware que lo ejecuta a código de máquina propio del dispositivo que lo ejecuta.
- Administrador de código: coordina toda la operación de los distintos subsistemas del *Common Language Entorno de ejecución.*
- Recolector de basura: elimina de memoria objetos no utilizados automáticamente.
- Motor de seguridad: administra la seguridad del código que se ejecuta.
- Motor de depuración: permite hacer un seguimiento de la ejecución del código aun cuando se utilicen lenguajes distintos.
- Verificador de tipos: controla que las variables de la aplicación usen el área de memoria que tienen asignado.
- Administrador de excepciones: maneja los errores que se producen durante la ejecución del código.
- Soporte de multiproceso (hilos): permite desarrollar aplicaciones que ejecuten código en forma paralela.

- Empaquetador de COM: coordina la comunicación con los componentes COM para que puedan ser usados por el .NET Framework.
- Biblioteca de Clases Base que incluye soporte para muchas funcionalidades comunes en las aplicaciones.

Ilustración 57 Estructura interna del entorno de ejecución en lenguaje común CLR.

En el SDK de Kinect para Windows se proporcionan APIs para código administrado y no administrado. La primera permite escribir programas en el lenguaje C++ o Visual Basic y la segunda en C#. Dadas la fiabilidad, facilidad de uso se usará la API de código gestionado en C# para la implementación de la Prueba de Concepto de este PFC.

3.5.1. Flujos de datos de imagen en Kinect NUI

Kinect for Windows SDK proporciona dos APIs, una nativa o no administrada para aplicaciones escritas en Visual Basic o C++ y una API administrada en C# para el desarrollo de aplicaciones que se beneficien del uso de las funcionalidades del sensor Kinect y que ejecuten en un entorno Windows.

Desarrollar una aplicación con Kinect para Windows es prácticamente el mismo proceso que desarrollar aplicaciones normales para Windows, con la diferencia que Kinect SDK ofrece al desarrollador soporte para el uso de las características de Kinect: imágenes en color, imágenes de profundidad, entrada de audio y seguimiento de esqueleto.

El SDK se compone de dos componentes principales: las librerías para el tratamiento de imagen y las librerías para el tratamiento de audio. Las primeras se han descrito en el apartado anterior, y las segundas se describen en el presente.

La API de NUI proporciona los medios para modificar la configuración de Kinect y para acceder a datos de imagen.

El flujo de datos se entrega como una sucesión de cuadros de imágenes fijas. Al inicializar la API de NUI, la aplicación identifica los flujos que utilizará. A continuación abre con esos flujos de datos con sus detalles de configuración específica: resolución, tipo de imagen, y número de búferes que el entorno de ejecución debe utilizar para almacenar imágenes entrantes. Si el entorno de ejecución llena todos los búferes antes de la aplicación adquiera una imagen (frame), el entorno de ejecución descarta el frame más antiguo y reutiliza el buffer. Como resultado, es posible que algunos frames sean descartados. Una aplicación puede solicitar hasta cuatro búferes, aunque dos búferes es suficiente para la mayoría de los escenarios de uso.

Una aplicación tiene acceso a los siguientes tipos de datos de imagen:

- Datos de imagen a color
- Datos de imagen de profundidad
- Datos de selección de jugador

3.5.1.1. Flujo de datos de imagen a color

Las imágenes en color están disponibles en los formatos siguientes:

- **RGB**
  El formato RGB de 32 bits proporciona mapas de bits de color lineales con formato X8R8G8B8, dentro del espacio de color sRGB. Para trabajar con los datos en formato RGB, una aplicación debe especificar un tipo de imagen RGB o color_YUV cuando se abre el flujo de datos.

- **YUV**
  El formato YUV proporciona mapas de bits lineales de 16 bits de color con corrección de gamma en formato de color UYVY, donde la corrección de gamma en el espacio de color YUV es equivalente a la gamma sRGB en el espacio RGB. Debido a que el flujo de YUV utiliza 16 bits por píxel, este formato utiliza menos memoria para almacenar los datos de mapa de bits y asigna menos memoria del búfer de transmisión cuando se abre el flujo de datos. Para trabajar con datos YUV, la aplicación debe especificar el tipo de imagen en bruto YUV cuando se abre el flujo. Los datos YUV sólo están disponibles en resolución 640 × 480 y sólo a 15 FPS.

Ambos formatos de color se calculan a partir de los mismos datos provenientes de la cámara, de modo que los datos YUV y los datos RGB representan la misma imagen. Por tanto, la elección de formato de datos debe ser la que sea más conveniente dada la implementación de la aplicación.

Un Kinect utiliza una conexión USB para transferir datos al PC, y esa conexión proporciona una cantidad limitada de ancho de banda. Los datos de color de imágenes de Bayer que el sensor vuelve a resolución 1280×1024 se comprimen y se convierten a formato RGB antes de la transmisión al entorno de ejecución. El entorno de ejecución a continuación descomprime los datos antes de que pase los datos a su aplicación. El uso de compresión hace que sea posible para devolver datos de color a velocidades de fotogramas tan alto como 30 FPS, pero el algoritmo que se utiliza conduce a cierta pérdida de calidad de imagen.

3.5.1.2. Flujo de datos de profundidad

El flujo de datos de profundidad proporciona frames en los que cada pixel contiene la distancia cartesiana (en milímetros) desde el plano de la cámara al objeto más cercano en las coordenadas X e Y respecto al campo de vista de la cámara de profundidad. Hay dos posibles rangos de datos de profundidad: el rango por defecto y el rango cercano. Estos valores están determinados por los valores de la enumeración *DepthRange*. Para elegir el formato de datos se utiliza la enumeración *DepthImageFormat*.

Las aplicaciones de usuario pueden procesar datos de un flujo de profundidad para dar soporte las funciones personalizadas, tales como el seguimiento de los movimientos de los usuarios y la identificación de otros objetos del escenario (muebles, paredes, etc.) para poder ignorarlos durante la ejecución del programa.

Cada píxel en el flujo de datos de profundidad utiliza 13 bits para representar la distancia y bits 3 para identificar al usuario. Un valor de 0 indica que no hay datos de profundidad disponible en esa posición, bien porque todos los

objetos estaban demasiado cerca de la cámara o porque estaban demasiado lejos de ella. Cuando el seguimiento de esqueleto está desactivado, los 3 bits que identifican a un usuario se ponen a 0.

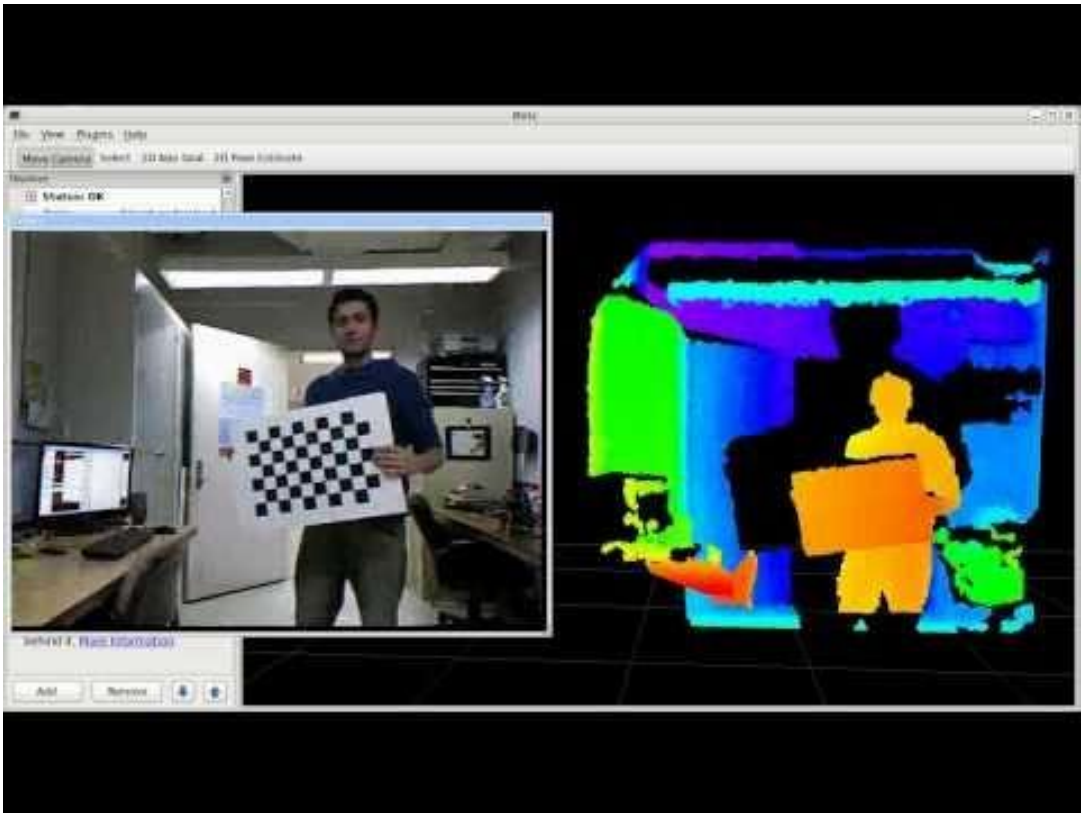


Ilustración 58 Ejemplo de flujos de datos de imagen a color y datos de profundidad (1)

Ilustración 59 Ejemplos de flujos de datos de imagen a color y datos de profundidad

Datos de identificación de usuarios

El sistema procesa los datos del sensor Kinect para identificar hasta seis figuras humanas y crea el mapa de la identificación del usuario. Este mapa es un mapa de bits en el que los valores de los píxeles corresponden al índice de usuario de la persona en el campo de visión que está más cerca de la cámara, en la posición del píxel. Los usuarios

pueden ser rastreados o no rastreados: sólo a los usuarios rastreados se les puede realizar un seguimiento del esqueleto completo con información de posición espacial de 20 articulaciones del cuerpo. Un máximo de 2 usuarios pueden ser rastreados al mismo tiempo mediante el sistema Kinect con las especificaciones actuales (en futuras versiones del sensor, se espera que esta funcionalidad sea ampliada). Las aplicaciones pueden elegir dos usuarios para realizar su seguimiento, o puede permitir que el sistema elija 2 por defecto.
Aunque la información de identificación de usuarios es una corriente lógica individual, en la práctica, los datos de profundidad y de identificación de datos de los usuarios se funden en una frame única:

- Los 13 bits de orden superior de cada píxel representan la distancia desde el sensor de profundidad al objeto más cercano, en milímetros.
- Los 3 bits de orden inferior de cada píxel representa el índice del usuario que es visible en las coordenadas *x* e *y* del píxel. Estos bits se tratan como un valor entero.

Un valor del índice de usuario de "0" indica que no se ha encontrado ningún jugador en ese lugar. Valores de "1" y "2" identifican a los usuarios. Las aplicaciones suelen utilizar los datos de identificación de usuarios como una máscara para aislar a los usuarios específicos o regiones de interés por el color y las imágenes en bruto de profundidad.

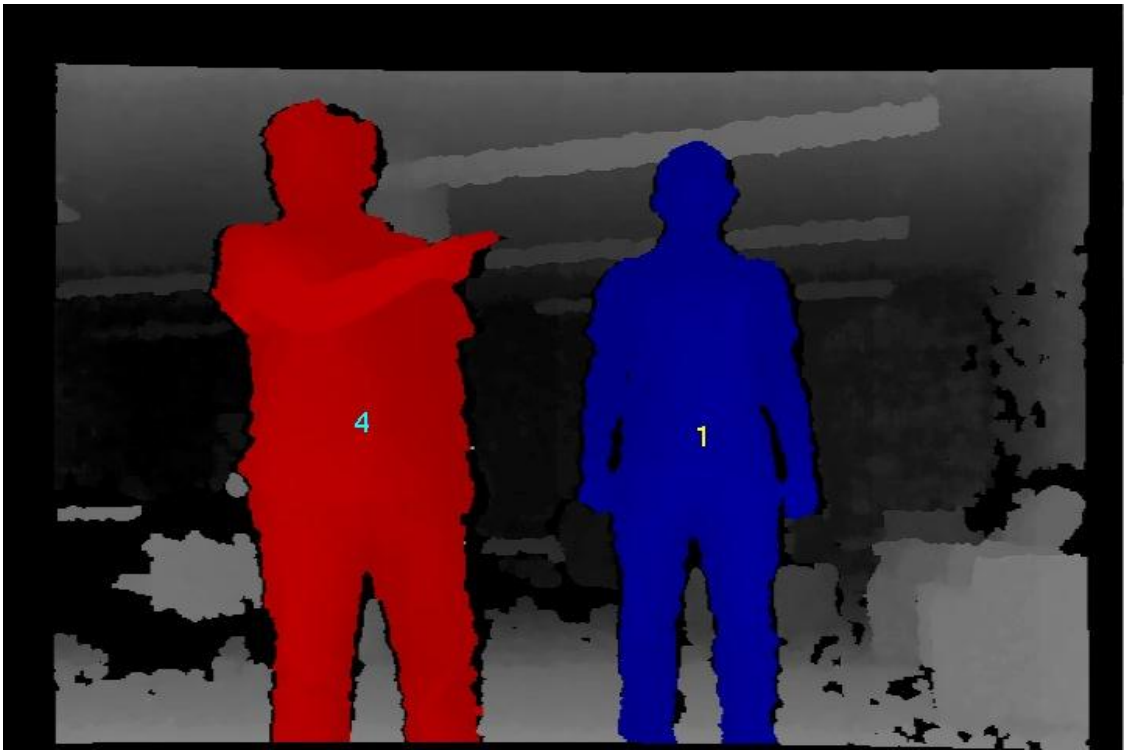


Ilustración 60 Identificación de usuarios a partir del flujo de datos de profundidad

3.5.1.3. Flujo de datos de seguimiento de esqueleto

El API de reconocimiento de esqueleto de la interfaz de usuario natural de Kinect puede realizar un seguimiento detallado de la posición de las articulaciones y la orientación para un máximo de dos usuarios al mismo tiempo.

Kinect puede ser consciente de hasta seis personas, y se puede realizar un seguimiento de dos de ellos en detalle. Para los dos esqueletos que se realiza un seguimiento, Kinect devuelve información completa de las 20 articulaciones.

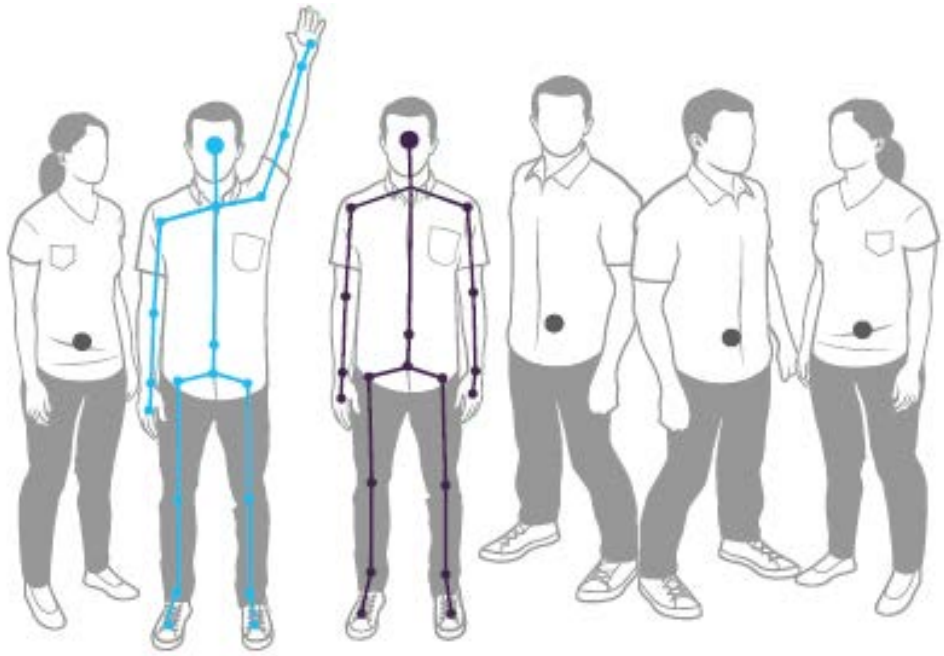


Ilustración 61 Seguimiento de esqueleto de Kinect

Los datos se proporcionan a la aplicación como un conjunto de puntos que componen un esqueleto, como se muestra en la siguiente figura. Este esqueleto representa la posición actual del usuario y la pose. Para utilizar los datos de seguimiento, una aplicación debe indicarlo tras inicializar el sensor habilitando el flujo de seguimiento de esqueleto.

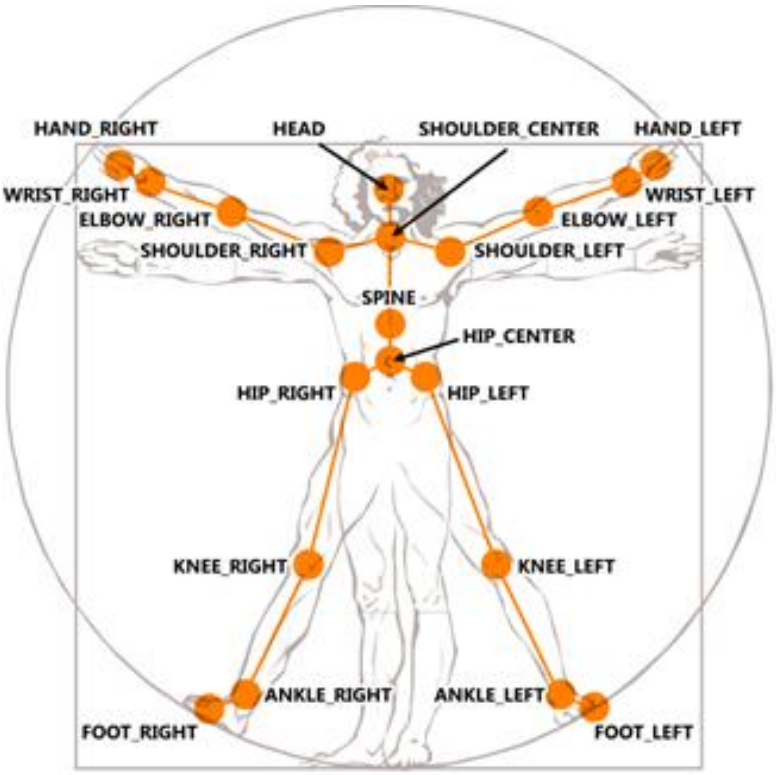


Ilustración 62 Skeleton positions relative to the human body

3.5.1.4. Obtención de datos de esqueleto del sensor

La aplicación obtiene la última información de datos de esqueleto de la misma manera que se consigue un frame de datos de imagen: llamando a un método de recuperación de frame y pasando un búfer. Las aplicaciones pueden usar ya sea un modelo de sondeo o un modelo basado en eventos, de la misma forma que ocurre para frames de imagen. Se debe elegir un modelo u otro, ya que una aplicación no puede utilizar los dos modelos al mismo tiempo.

Para utilizar el modelo de sondeo:

- C++ (código no administrado) ejecuta el método *INuiSensor::NuiSkeletonGetNextFrame* para obtener un frame de esqueleto.
- C# (código administrado) ejecuta el método *SkeletonStream.OpenNextFrame*.

Para utilizar el modelo de eventos:

- El código C++ pasa un controlador de eventos al método *INuiSensor::NuiSkeletonTrackingEnable*. Cuando un nuevo frame de datos de esqueleto está listo, el evento se lanza. Cualquier hilo que estuviera esperando despierta y obtiene la los datos de esqueleto llamando al método *INuiSensor::NuiSkeletonGetNextFrame*. Durante este tiempo, el evento ha sido reiniciado por la API.
- El código C # utiliza el modelo de eventos asignando el evento *KinectSensor.AllFramesReady* a un controlador de eventos adecuado.

Cuando un nuevo frame de datos de esqueleto está listo, el evento está activo, el controlador del evento se ejecuta y se llama a la función *NuiSkeletonGetNextFrame* para obtener el frame.

El motor de seguimiento esquelético procesa los datos de profundidad de imagen para calcular el plano del suelo, (la superficie sobre la que están de pie los usuarios). Si la aplicación indica en el momento de inicialización que requiere la funcionalidad de seguimiento de esqueleto, el motor de seguimiento esquelético envía un frame de datos de esqueleto cada vez que procesa los datos de profundidad, aparezca o no un esqueleto en el frame. Las aplicaciones que utilizan los valores de recorte de piso plano puede recuperar la estructura del esqueleto.

3.5.1.5. Seguimiento de esqueleto Activo y Pasivo

La funcionalidad de seguimiento del esqueleto proporciona un seguimiento completo del esqueleto de uno o dos usuarios dentro del campo de visión de las cámaras. Cuando un usuario está siendo seguido activamente, las llamadas para obtener los siguientes datos de esqueleto devolverán información completa de esqueleto para el usuario activo.

El seguimiento pasivo se proporciona de forma automática para hasta cuatro usuarios adicionales en el campo de visión. Cuando a un usuario se le hace un seguimiento de forma pasiva, el frame de datos de esqueleto contiene sólo información limitada a cerca de la posición de ese usuario. Por defecto, los primeras dos esqueletos que el sistema de seguimiento del esqueleto encuentra son rastreados de forma activa, como se muestra en ilustración 64.

El entorno de ejecución devuelve datos esqueleto en un frame de esqueleto, el cual contiene una serie estructuras de datos sobre esqueletos, una para cada esqueleto que el sistema ha reconocido. No todos los marcos esqueleto contienen datos sobre el esqueleto. Cuando el seguimiento esqueleto está habilitado, el entorno de ejecución activa un evento de esqueleto cada vez que procesa una imagen/frame de profundidad, como se describe en la sección anterior.

Para todos los esqueletos de retorno, se proporcionan los siguientes datos:

- El estado actual de seguimiento del esqueleto asociado:
  - Para esqueletos que están siendo rastreados pasivamente, este valor sólo indica que se registra información simple de posición.
  - Para un esqueleto de un seguimiento activo, el valor indica que se recoge seguimiento de esqueleto.

- Un identificador de seguimiento único que queda asignado a un solo usuario mientras éste se desplaza por el campo de visión.

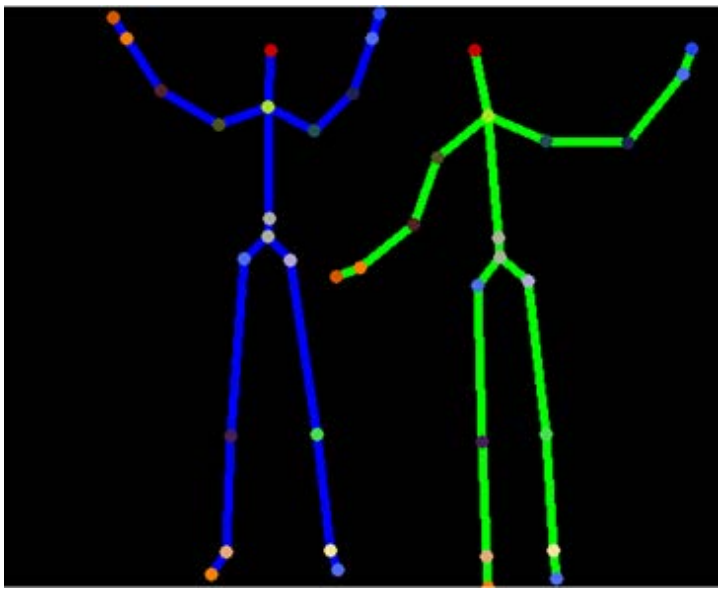


Ilustración 63 Active tracking for two players

Se garantiza que el identificador de seguimiento permanecerá constantemente asignado al mismo usuario durante el tiempo que éste permanece en el campo de visión de Kinect. Un identificador de seguimiento dado se garantiza que permanecen en el mismo índice en la matriz de datos de esqueleto durante tanto tiempo como el identificador de seguimiento está en uso. Si el ID de seguimiento del esqueleto cambia, o bien el usuario al que se le realiza el seguimiento ha abandonado el campo de visión y el sensor detectó otro usuario en el campo de visión, o bien el usuario dejó el campo de visión, y luego regresó y se reanudó el seguimiento.

- Una posición (de tipo Vector4) indica el centro de la masa para ese usuario. Este valor es el único valor calculado respecto a la posición de usuarios pasivos.
- Para los usuarios con seguimiento activo, se incluyen también los datos actualizados del esqueleto completo.
- Para los usuarios en seguimiento pasivo, los datos devueltos incluyen sólo la posición del centro de masa y los datos de identificación, pero no los datos del esqueleto.

3.5.2. Flujo de datos de audio

Para el objetivo de este documento se van a analizar con más detalle las librerías del SDK enfocadas al tratamiento de audio capturado usando el array de cuatro micrófonos integrado en el sensor Kinect. Al igual que sucedía con la librería NUI para el tratamiento de imágenes y datos de profundidad, existe un API administrada para C# y otra no administrada para C++. En este apartado se va a analizar únicamente la API administrada al haber sido la utilizada en el análisis de sensibilidad y en la realización de la prueba de concepto.

Introducción

El conjunto de micrófonos es la joya escondida del sensor Kinect. La matriz está constituida por cuatro micrófonos separados que se extienden linealmente en la parte inferior del cabezal de Kinect. Al comparar el instante temporal en el que cada micrófono captura la misma señal de audio, la matriz de micrófonos se puede utilizar para determinar la dirección de la que la señal está llegando. Esta técnica también se puede utilizar para hacer que la matriz de micrófono filtre el sonido proveniente de una dirección particular en lugar de otra. Por último, se pueden aplicar complejos al conjunto de micrófonos con el fin de realizar efectos para eliminar el ruido de fondo irrelevante. Esto permite que se utilicen comandos de voz en una gran sala donde el jugador siempre que el jugador esté situado a más de unos centímetros del micrófono.

Cuando Kinect fue lanzado por primera vez para la Xbox 360, el conjunto de fue dejado en segundo plano. Esto se debió en parte a la emoción sobre el seguimiento de esqueleto, lo que parecía una tecnología mucho más innovadora, pero también en parte a que los esfuerzos para utilizar las funciones de audio del sensor se dedicaron al panel de control gestionado por voz de Xbox.

Fundamentos del array de micrófonos

Al instalar el SDK de Kinect, los componentes necesarios para el reconocimiento de voz son automáticamente instalados. El conjunto de micrófonos Kinect trabaja en la parte superior del código ya existente en librerías que se incluyen en el sistema operativo desde Windows Vista. Estos componentes prexistentes incluyen *Voice Capture DirectX Media Object* (DMO) y *Speech Recognition API* (SAPI).

El componente Voice Capture DMO está destinado a proporcionar un API para trabajar con conjuntos de micrófonos y de esta forma proporcionar la funcionalidad de cancelación de eco (AEC), control automático de ganancia (AGC), y supresión de ruido. Esta funcionalidad se puede encontrar en las clases de audio del SDK. La envoltura de Kinect SDK de audio simplifica el trabajo con la DMO, al mismo tiempo que optimiza el rendimiento de DMO con el sensor.

Para que Kinect pueda aplicar el reconocimiento de voz usando el SDK, la siguiente librerías son necesarias: API y SDK de reconocimiento de voz (*Speech Recognition API y SDK*), y el paquete de idioma de Kinect para Windows (*Kinect for Windows Runtime Language Pack*).

La API de voz es simplemente una biblioteca de desarrollo que extiende las funcionalidades incorporadas en el sistema operativo para reconocimiento de voz. Se puede utilizar con o sin el SDK de Kinect. Por ejemplo, si se desea agregar comandos de voz para una aplicación de escritorio estándar que use un micrófono que no sea el array de micrófonos.

El paquete de idioma de Kinect para Windows, por el contrario, es un caso conjunto especial de modelos lingüísticos utilizados para la interoperabilidad entre el SDK de Kinect y los componentes de SAPI. Así como el reconocimiento de esqueleto de Kinect requiere de un modelado computacional masivo para entrenar los árboles de decisión para interpretar posiciones de puntos de unión, la biblioteca SAPI requiere de un complejo diseño para ayudar en la interpretación de los patrones de lenguaje según son recibidos por el conjunto de micrófonos Kinect. El paquete de idioma Kinect ofrece estos modelos para optimizar el reconocimiento de comandos de voz.

La clase principal para trabajar con audio en el API administrado para C# se denomina *KinectAudioSource.* El propósito de la clase *KinectAudioSource* es transmitir ya sea audio en bruto o modificado a partir de la matriz de micrófonos. El flujo de audio puede ser modificado para incluir una variedad de algoritmos para mejorar su calidad. Entre estos algoritmos se incluyen: supresión de ruido, control automático de ganancia (AEC) y cancelación de eco acústico. *KinectAudioSource* se puede utilizar para configurar el array de micrófonos para trabajar en diferentes modos. También puede ser utilizado para detectar la dirección de dónde proviene principalmente la fuente de audio, así como para forzar al conjunto de micrófonos de filtrar el audio proveniente de una determinada dirección.

Con el fin de trabajar con el *KinectAudioSource* es útil familiarizarse con parte del vocabulario utilizado en la transmisión y grabación de audio. El siguiente glosario debe usarse como una referencia para entender conceptos abstractos usados por la clase *KinectAudioSource.*

- **Cancelación de eco** o **Acoustic Echo Cancellation** (AEC) se refiere a una técnica para hacer frente a ecos acústicos. Estos ecos ocurren cuando el sonido de un altavoz es recibido por micrófono. Una manera de entender esto es pensar en lo que sucede cuando se usa un teléfono y se oye la propia voz con un cierto grado de retraso, que se repite en el receptor. AEC trata con este problema restando los patrones de sonido que emite un altavoz del sonido recogido por el micrófono.
- **Supresión del eco acústico** o **Acoustic Echo Suppression** (AES) se refiere a algoritmos utilizados para eliminar el eco residual que queda después de que se ha realizado AEC.
- **Control Automático de Ganancia** o **Automatic Gain Control** (AGC) se refiere a algoritmos utilizados para hacer que la amplitud de la voz del interlocutor sea constante en el tiempo. Cuando éste se aproxima a un altavoz o se aleja del micrófono, su voz puede convertirse en más fuerte o más suave. AGC trabaja para igualar estos cambios.
- **Formación del haz n** o **Beamforming** se refiere a técnicas algorítmicas que emulan un micrófono direccional. Este punto se describirá en detalle en el siguiente apartado.
- **Supresión del ruido** o **Noise Suppression** (NS) se utiliza para eliminar patrones que no son propios de voz humana de la señal recibida por los micrófonos del array. Al eliminar el ruido de fondo, el audio recogido por el micrófono se hace más limpio y claro.

La clase *KinectAudioSource* ofrece un alto nivel de control sobre muchos aspectos de la grabación de audio, a pesar de que actualmente no expone todos los aspectos de la DMO subyacente. Las diversas propiedades utilizadas para modificar el procesamiento de audio capturado por el sensor y controlado a través de *KinectAudioSource* se denominan características o *features*.

La siguiente tabla muestra el las características que pueden ser ajustadas:

Name	Valores / Por defecto	Descripción
AutomaticGainControlEnabled	True, False Default: False	Activa o desactiva el algoritmo de control de ganancia automática del DMO.
BeamAngleMode	Adaptive Automatic Manual Default: Automatic	Especifica qué algoritmo usar para el procesamiento del audio capturado.
EchoCancellationMode	CancellationAndSuppression CancellationOnly None Default: None	Activa o desactiva AEC. Puede activarse exclusivamente AEC o activar AEC con AES para señales residuales
NoiseSuppression	True, False Default: False	Especifica si se ejecuta el algoritmo de supresión de ruido

Tabla 1 Características ajustables de Kinect Audio

BeamAngleMode abstrae el modo de operación del sistema subyacente DMO y del conjunto de micrófonos. En el nivel de DMO se determina si la DMO debe gestionar el proceso de *beamforming* o delegarlo a la aplicación. Además, el SDK de Kinect para Windows proporciona un conjunto adicional de algoritmos para llevar a cabo la formación de haz, delegando de esa forma ese trabajo al SDK. La siguiente tabla describe los posibles valores de *BeamAngleMode.*

Name	Descripción
Adaptive	Beamforming controlada por algoritmos creados para Kinect SDK
Automatic	Beamforming controlada por el DMO
Manual	Beamforming controlado por la aplicación

Tabla 2 Enumeración de posibles valores de BeamAngleMode

Beamforming en modo Adaptativo aprovechará de las características peculiares del sensor Kinect para encontrar la fuente de sonido correcta del mismo modo que el seguimiento de esqueleto trata de encontrar la persona correcta para realizar el seguimiento. Al igual que con el seguimiento de esqueleto, el proceso de formación de haz de Kinect también se puede poner en modo manual, lo que permite a aplicaciones solicitar la dirección en que quieren concentrarse en el sonido. Para utilizar el sensor Kinect como un micrófono direccional, es necesario configurar el modo a Manual y proporcionar un valor de la propiedad *ManualBeamAngle* de la clase *KinectAudioSource*.

3.6. Nuevas funcionalidades de *Kinect SDK for Windows 1.5*

Aunque este PFC esté centrado en las funcionalidades de tratamiento de audio del sensor Kinect precisamente para no caer en la tentación de realizar un trabajo basado en el uso estrella de Kinect, el sensor de profundidad, la última versión del SDK de Kinect para Windows incorpora dos nuevas funcionalidades inéditas e innovadoras. Estas funcionalidades se incorporan sobre el mismo hardware de Kinect y son el **seguimiento facial** y el **seguimiento de esqueleto de cuerpo sentado**.

La funcionalidad de **reconocimiento facial** permite usar la cámara de profundidad para identificar una malla en tres dimensiones del rostro de una persona situada a corta distancia del sensor. La principal consecuencia es que a partir de ahora es posible **reconocer gestos y expresiones faciales** de un individuo, así como obtener características fisiológicas como la posición de los ojos, de la boca, etc. Podría usarse con un poco de trabajo en identificación de personas, animación de avatares,…

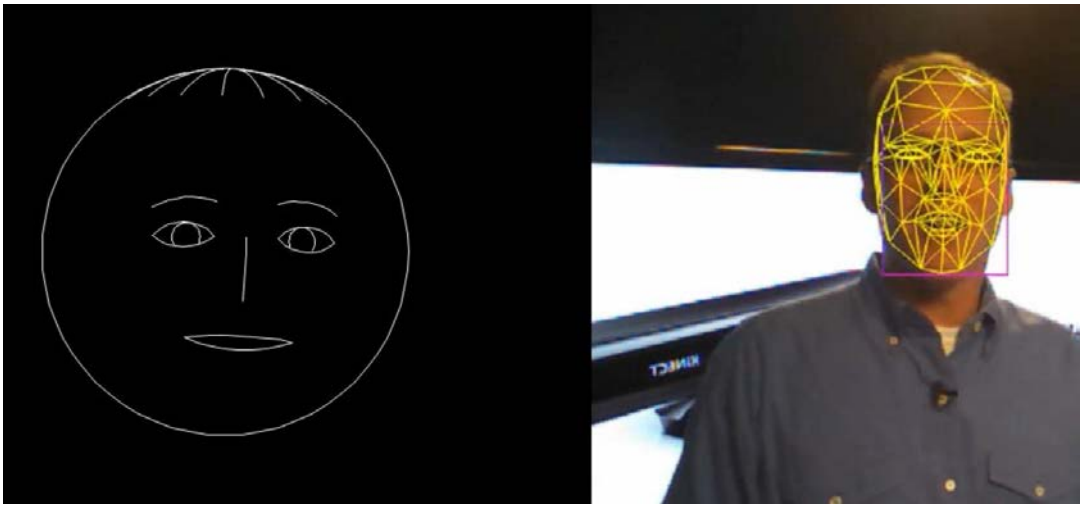


Ilustración 64 Reconocimiento facial usando Kinect (1)

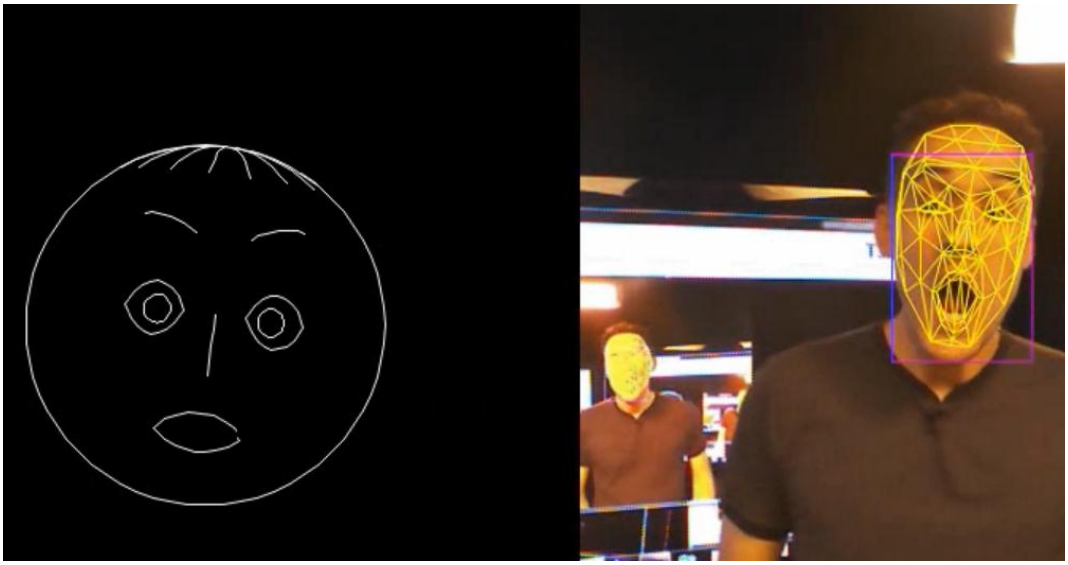
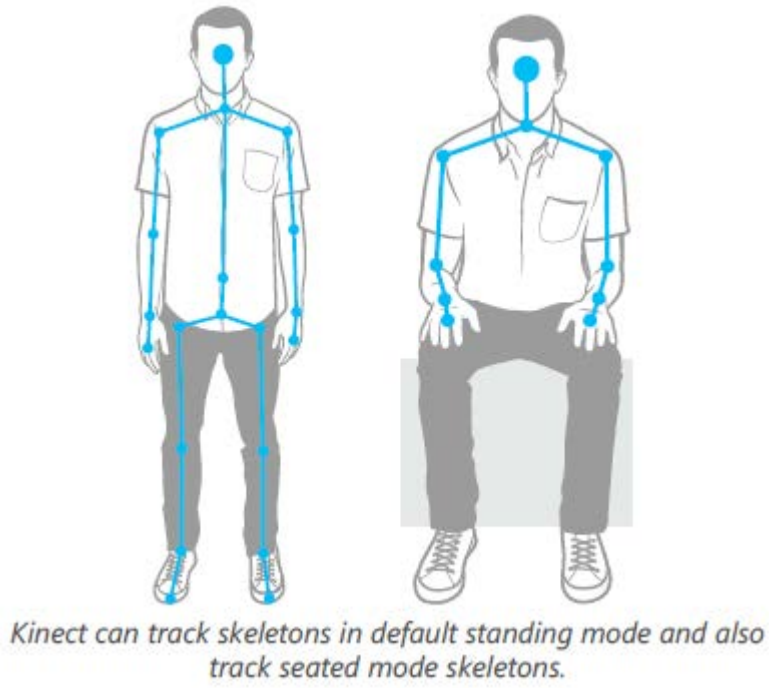


Ilustración 65 Reconocimiento facial usando Kinect (2)

La otra gran novedad es la mejora del sistema se **seguimiento de esqueleto**. Ahora será posible realizar el seguimiento de un usuario situado en el campo de visión de Kinect incluso si éste se encuentra sentado (recordemos que hasta esta versión del SDK, únicamente era posible realizar seguimiento de esqueleto de cuerpo completo).

Esta nueva funcionalidad usa el Modo Cercano de Kinect para representar un esqueleto de usuario en posición sentada usando para ello un conjunto de 10 articulaciones de modo que sea posible usarlo para realizar control preciso de control de gestos basados en movimientos de las articulaciones de mano y brazo.



Kinect can track skeletons in default standing mode and also track seated mode skeletons.

Ilustración 66 Esquemas de reconocimiento de esqueleto de Kinect

Como pasa siempre con Kinect, las posibilidades son (casi) incontables.

Desarrolladores, **la pelota está en nuestro tejado.**

3.7. Guía de programación con Kinect usando C#

Este apartado sirve a modo de manual de usuario sobre programación con Kinect a nivel introductorio usando el SDK de Kinect para Windows.

3.7.1. Instalación

Para poder programar usando Kinect es necesario instalar en el sistema el SDK de Kinect, que se encuentra en la página web oficial de Kinect for Windows. Simplemente hay que descargarse la versión dependiendo de nuestro sistema operativo (32 o 64 bits) e iniciar el fichero ejecutable.

De este modo se instalarán las librerías y controladores necesarios para interactuar con Kinect.

A continuación, se crea un nuevo proyecto en Microsoft Visual Studio 2010. Dependiendo del objetivo del proyecto, éste puede ser WPF si se desea una aplicación de escritorio estándar, o bien puede ser una aplicación de consola, un formulario Forms, etc.

Lo importante es que una vez esté creado el proyecto hay que añadir las referencias a las librerías del SDK para para poder usarlas en nuestro proyecto. Para añadirlas, simplemente se selecciona en el panel denominado "Soluciones", sobre la carpeta "Referencias", hacer click secundario y seleccionar "Añadir referencia":

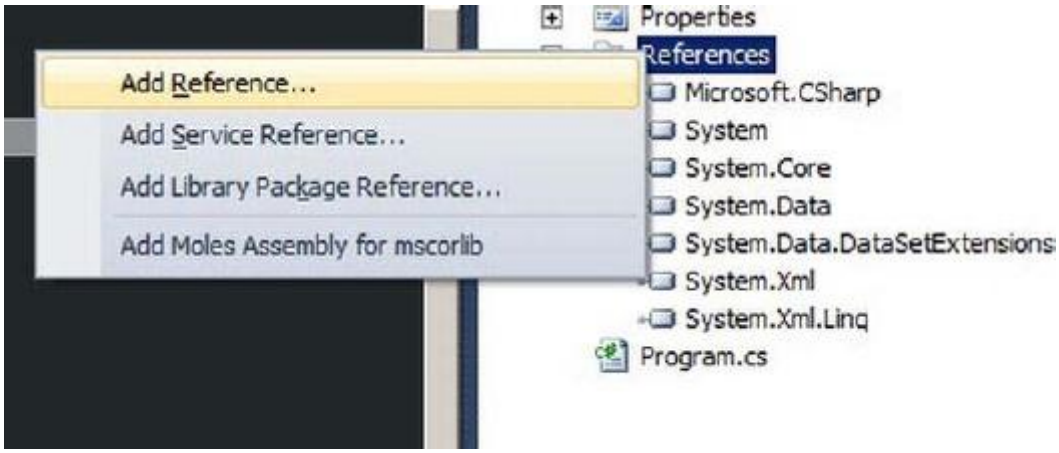


Ilustración 67 Añadir referencia a un proyecto

En la ventana que aparece se listan las bibliotecas que se pueden usar en el proyecto. Bajo la pestaña .NET, seleccionar la denominada "Microsoft.Kinect".

Ahora, al comienzo del código principal (por defecto *Program.cs* para Forms o *MainWindow.cs* para WPF) se añade la referencia a esta biblioteca:

```
using Microsoft.Kinect;
```

Con esta sencilla configuración previa ya estamos listos para desarrollar aplicaciones usando el SDK de Kinect.

3.7.2. Inicialización del sensor

Tres acciones son estándar para las aplicaciones de Kinect que se aprovechan de los flujos de datos ofrecidos por Kinect: El objeto KinectSensor debe ser creado, inicializado, y arrancado a continuación.

A continuación se muestra un programa sencillo en código administrado C# en el que se realiza este proceso:

```
static void Main(string[] args){
       // instantiate the sensor instance
       KinectSensor sensor = KinectSensor.KinectSensors[0];

       // initialize the cameras
       sensor.ColorStream.Enable();
       sensor.ColorFrameReady += sensor_ColorFrameReady;

       //for depth: sensor.DepthStream.Enable();
       //            sensor.DepthFrameReady += sensor_DepthFrameReady;

       // make it look like The Matrix
       Console.ForegroundColor = ConsoleColor.Green;

       // start the data streaming
       sensor.Start();
       while (Console.ReadKey().Key != ConsoleKey.Spacebar) { }
}
```

Los pasos son los siguientes:

- Se crea e inicia el objeto que representa al sensor Kinect. En este caso se asume que sólo hay una cámara.
- Se activan los flujos de datos que se deseen usar. En este caso se activa la cámara RGB.
- Se declara el manejador de eventos para el evento `ColorFrameReady`.
- Se entra en un bucle infinito que acaba cuando el usuario presiona la tecla espaciadora.

Este será uno de los primeros pasos en cualquier aplicación que use Kinect, ya que lo primero es declarar el sensor, activar los flujos de datos y activar el sensor. Este paso suele ser bastante repetitivo entre programas, y donde realmente se encuentra la funcionalidad es en los manejadores de eventos: `ColorFrameReady`, `DepthFrameReady` y `SkeletonFrameReady`.

Es en estas funciones donde se describirá lo que se desea hacer con los flujos de datos.

3.7.3. Gestión de flujos de datos de imagen

En este apartado se describirán sencillas operaciones que pueden realizarse con los flujos de datos de imagen obtenidos del sensor Kinect.

Flujos de datos de la cámara RGB

Primero, habilitamos el flujo de datos de color desde la cámara RGB:

```
this._KinectDevice.ColorStream.Enable();
this._KinectDevice.ColorFrameReady += KinectDevice_ColorFrameReady;
```

El siguiente ejemplo muestra cómo mostrar el flujo de datos de la cámara RBG en un control de usuario de la interfaz principal de usuario mediante el manejador del evento `ColorFrameReady` .

```
private void Kinect_ColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
{
   using(ColorImageFrame frame = e.OpenColorImageFrame())
   {
      if(frame != null)
      {
          byte[] pixelData = new byte[frame.PixelDataLength];
          frame.CopyPixelDataTo(pixelData);
          ColorImageElement.Source = BitmapImage.Create(frame.Width, frame.Height,
                                          96, 96,PixelFormats.Bgr32, null,
                                          pixelData,frame.Width * frame.BytesPerPixel);
      }
   }
}
```

El procesamiento de los datos de imagen comienza por obtener o abrir el frame o imagen. El método `OpenColorImageFrame` del objeto `ColorImageFrameReadyEventArgs` devuelve el objeto actual ColorImageFrame que representa la última imagen capturada por la cámara RGB del sensor. La extracción de datos de los píxeles del fotograma requiere primero crear un array de bytes para almacenar los datos.

La propiedad `PixelDataLength` en el objeto frame informa del tamaño exacto de los datos y, consecuentemente, el tamaño del array. Llamar a la función `CopyPixelDataTo` rellena el array con los datos de los píxeles. La última línea de código crea una imagen de mapa de bits de los datos de píxeles y representa la imagen en la interfaz de usuario. En este caso `ColorImageElement` es un objeto de tipo Image que representa un control de usuario para mostrar imágenes en la IU.

Flujos de datos del sensor de profundidad

Al igual que antes, primero se inicializa el sensor habiendo declarado que se va a usar el sensor de profundidad:

```
this._KinectDevice.DepthStream.Enable();
this._KinectDevice.DepthFrameReady += KinectDevice_DepthFrameReady;
```

Un manejador del evento `DepthFrameReady` sencillo aparece a continuación.

```
private void Kinect_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs e)
{
 using(DepthImageFrame frame = e.OpenDepthImageFrame())
 {
 if(frame != null){
 short[] pixelData = new short[frame.PixelDataLength];
 frame.CopyPixelDataTo(pixelData);
 int stride = frame.Width * frame.BytesPerPixel;
 DepthImage.Source = BitmapSource.Create(frame.Width, frame.Height,
         96, 96 ,PixelFormats.Gray16, null,pixelData, stride);
 }
}
```

Cuando Kinect tiene una nueva imagen de profundidad disponible para procesar, el `KinectSensor` desencadena el evento `DepthFrameReady`. Este controlador de eventos simplemente toma los datos de la imagen y crea un mapa de bits, que luego se muestran en la ventana de la interfaz de usuario. La siguiente ilustración es un ejemplo del flujo de datos de profundidad. En ella se observa cómo los objetos situados cerca de Kinect son representados en una escala de color gris o negro. Cuanto más lejos está un objeto de Kinect, más claro será el gris.

Ilustración 68 Flujo de datos de profundidad

3.7.4. Localización de fuente acústica

Este apartado describe ejemplos en C# para usar la funcionalidad de audio de Kinect para calcular la dirección de la que proviene una señal acústica.

Habilitar flujo de datos de audio

Antes de nada, es preciso inicializar el flujo de datos del sensor. En el siguiente ejemplo se muestra un ejemplo de cómo hacerlo. Es necesario tener en cuenta que el flujo de audio tarda cuatro segundos en activarse. Este proceso se puede realizar, por ejemplo, con un objeto temporizador como el siguiente:

```
// NOTE: Need to wait 4 seconds for device to be ready to stream audio right after
initialization
this.readyTimer = new DispatcherTimer();
this.readyTimer.Tick += this.StartAudioStream;
this.readyTimer.Interval = new TimeSpan(0, 0, 4);
this.readyTimer.Start();
```

Al cabo de cuatro segundos se ejecuta la función `StartAudioStream`:

```
/// <summary>
/// Starts the audio stream
/// </summary>
private void StartAudioStream()
{
       var audioSource = this.kinectSensor.AudioSource;
       audioSource.BeamAngleMode = BeamAngleMode.Adaptive;
    this.audioStream = audioSource.Start();
}
```

Tras ello, se activa el flujo de datos de audio. En este caso se ha establecido el modo de funcionamiento de *beamforming* a Adaptativo.

Localizar fuente de sonido

Al iniciar el flujo de datos, el sensor inicia automáticamente el proceso de localización de fuente sonora. Hay dos formas de acceder a estos datos en continuo cambio.

1. Mediante sondeo
   Cada vez que el desarrollador lo desee, puede obtener los valores del ángulo concreto calculado y el grado de confianza estimado, accediendo a los atributos `SoundSourceAngle` y `SoundSourceAngleConfidence` de la clase KinectAudioSource, respectivamente. Para obtener el rango más general de la dirección de origen de sonido (*beam*), se usa el atributo `BeamAngle`.
   El valor del ángulo calculado entra dentro del tango [-50, 50] grados, la confianza tiene el rango [0, 1] siendo 1 confianza plena en el valor calculado, y el valor del área (*beam*) de origen es un redondeo del ángulo concreto: Si el ángulo es -17,2 grados, se devolverá -20 grados; si el ángulo es 34 grados, se devuelve 30 grados.

2. Mediante eventos:
   Este es el método más usado. Simplemente se define un manejador de eventos asignado a uno de los dos eventos que el SDK controla para el cambio en la dirección de una fuente sonora. Uno tiene que ver con el valor concreto del ángulo (`SoundSourceAngle`) y otro con el área (`BeamAngle`). Por regla general se lanzarán muchos más procesos de cambio de `SoundSourceAngle` debido a que su valor es mucho más preciso (hasta dos decimales) y la velocidad de refresco es muy baja (del orden de décimas de segundo).

Los eventos que controlan estos cambios son `BeamAngleChanged` y `SoundSourceAngleChanged`. Para asignarles un manejador de evento, por ejemplo:

```
this.sensor.AudioSource.BeamAngleChanged += this.AudioSourceBeamChanged;
this.sensor.AudioSource.SoundSourceAngleChanged += this.AudioSourceSoundSourceAngleChanged;
```

De esta forma, cada vez que el sensor detecte que ha habido un cambio o bien en el valor de ángulo calculado o en el *beam,* lanzará el evento correspondiente y el programa tratará este evento como desee.

4. Capítulo 4: Análisis de sensibilidad

La localización de fuente sonora es la principal funcionalidad de audio de Kinect y será la que se use en la prueba de concepto. Para poder realizar un análisis sobre esta característica, es necesario primero explicar algunos conceptos teóricos que se van a utilizar y que no son comúnmente conocidos. En el primer apartado de este capítulo se describen estos conceptos.

En el segundo apartado se describen las pruebas que se han realizado para probar el rendimiento del sensor con el objetivo de estudiar qué parámetros influyen en la calidad y precisión de Kinect a la hora de calcular la dirección de una fuente sonora activa.

4.1. Definiciones previas

La matriz de micrófonos de Kinect es un sofisticado sensor que, dadas sus características, es capaz de detectar la dirección de una fuente sonora, realizar complejos algoritmos sobre una fuente de audio para suprimir ruido, echo y controlar la estabilidad de la señal de voz acústica humana. Todo esto no sería posible con un micrófono convencional, y es por esto que en este apartado se desea describir con detalle tanto el conjunto de micrófonos incorporado en Kinect como los fundamentos teóricos en los que se basan los algoritmos de formación de haz o *beamforming.*

4.1.1. Array de micrófonos

Un **array, conjunto o matriz de micrófonos** es cualquier número de micrófonos que funcionan en tándem. Tienen diversas aplicaciones:

- Sistemas de extracción de señales de voz de una fuente contaminada con ruido ambiental (por ejemplo, teléfonos, sistemas de reconocimiento de voz, audífonos).
- Tecnología de sonido envolvente y similar.
- Localización de objetos a partir del sonido que producen: localización de la fuente acústica. Usos militares son, por ejemplo, la localización una fuente de fuego de artillería o localización de aviones y seguimiento.
- Grabaciones de alta fidelidad.

Un array de 1020 micrófonos, el más grande del mundo, fue construido por los investigadores del *MIT Computer Science and Artificial Intelligence Laboratory*.

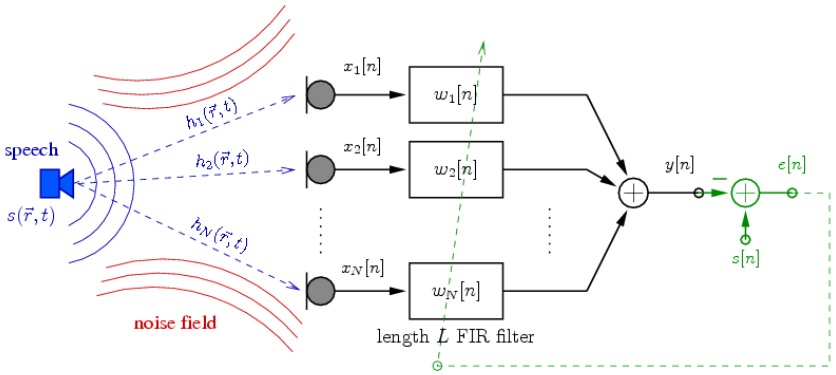


Fig. 1: Multi-channel adaptive system

Ilustración 69 Algoritmo de supresión de ruido usando un array de micrófonos

Microsoft Windows Vista y superiores sistemas operativos soportan el uso de arrays de micrófonos para aumentar la exactitud de su función de reconocimiento de voz, permitiendo a los usuarios conectar varios micrófonos a un solo sistema, de modo que las entradas se pueden combinar en una sola, de mayor calidad de la fuente.

Las características de audio de Kinect están soportadas por un conjunto o array de cuatro micrófonos que están separados varios centímetros y dispuestos en un patrón lineal. Una matriz de múltiples micrófonos independientes tiene las siguientes ventajas significativas sobre un único micrófono:

- Su superior calidad de sonido permite que los algoritmos de supresión del ruido y cancelación de eco acústico (AEC) sean más sofisticados y eficaces que los algoritmos posibles con un solo micrófono.
- Permiten implementar técnicas de *beamforming* y localización de origen de sonido. Usando el hecho de que el sonido procedente de una fuente de audio en particular llega a cada micrófono del array en un tiempo ligeramente diferente, la técnica de *beamforming* permite que las aplicaciones conozcan la dirección de la fuente de audio y utilizar la array de micrófonos como un micrófono direccional orientable.

El array lineal de cuatro micrófonos de Kinect utiliza un conversor analógico-digital de 24 bits y proporciona procesamiento local de la señal, con eliminación de eco y supresión de ruido. Las aplicaciones que se crean con este SDK pueden utilizar la matriz de micrófono de Kinect para:

- Captura de audio de alta calidad.
- *Beamforming* y la localización de fuente de audio. KinectAudio incluye algoritmos que controlan el haz o *beam* de sonido y proporcionan la dirección de la fuente a las aplicaciones.
- Reconocimiento de voz.

Con *Kinect for Windows SDK*, las aplicaciones pueden utilizar el conjunto de micrófonos Kinect como un dispositivo de entrada para el reconocimiento del habla junto con el API Microsoft Speech.

4.1.2. Beamforming

Se trata de una técnica de procesamiento de señal utilizado en conjuntos de sensores para la transmisión o recepción de una señal con una dirección determinada. Esto se consigue mediante la combinación de elementos del conjunto de sensores de tal manera que las señales en determinados ángulos experimenten interferencia constructiva mientras que otros ángulos experimenten interferencia destructiva. La técnica de *Beamforming* o formación de haz puede ser utilizada tanto en el lado de transmisión como en el de recepción con el fin de conseguir una selectividad espacial. La mejora en comparación con la recepción/transmisión omnidireccional se conoce como la ganancia de recepción.

Esta técnica puede ser utilizada para las ondas de radio o de sonido. Se la han encontrado numerosas aplicaciones en dispositivos como el radar, el sónar, y en campos como la sismología, las comunicaciones inalámbricas, la radioastronomía, la acústica, y la biomedicina. La variante de *Adaptive Beamforming* (beamforming adaptativa) se utiliza para detectar y estimar la señal de interés en la salida de un conjunto de sensores por medio de datos de filtrado espacial de adaptación y el rechazo de interferencia.

Esta técnica está implementada en Kinect, y por medio de las APIs de audio seremos capaces de obtener una fuente de audio proveniente de una zona espacial específica ignorando el resto.

Técnicas de beamforming

Beamforming se aprovecha de la interferencia de señales para cambiar la direccionalidad de la matriz. Cuando se transmite, un emisor controla la fase y la amplitud relativa de la señal en cada transmisor, con el fin de crear un

patrón de interferencia constructiva y destructiva en el frente de onda. Cuando se está recibiendo, la información de diferentes sensores se combina de forma que el patrón de radiación esperado es observado preferentemente.

Por ejemplo, en un sónar, para enviar un pulso de sonido debajo del agua hacia un barco a distancia, si se transmite ese pulso desde todos los proyectores del array de sónar del barco al mismo tiempo el proceso falla ya que el barco oye primero el pulso por el altavoz situado más cerca del barco y más tarde por los situados a más distancia de la nave. La técnica de *beamforming* consiste en enviar el pulso de cada emisor en momentos de tiempo ligeramente diferentes (el emisor más cercano emite en última posición), de modo que cada pulso llega a la nave destino a la vez, produciendo el efecto de un fuerte pulso único desde un único emisor. El mismo proceso puede llevarse a cabo en aire usando altavoces de gran potencia, o usando radares o radios por medio de antenas.

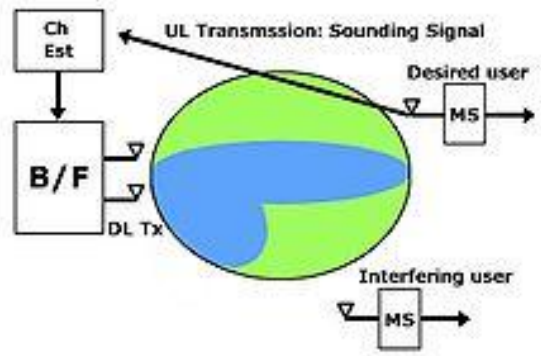


Ilustración 70 Beamforming (1)

En un sónar pasivo, y en la recepción de un sónar activo, la técnica de *beamforming* implica la combinación de señales retardadas de cada hidrófono en momentos de tiempo ligeramente distintos (el hidrófono más cercano al destino/objetivo se combinará con un retraso más largo), de manera que cada señal alcance la salida de origen exactamente al mismo tiempo, produciendo una única señal fuerte, como si la señal proviniese de un hidrófono único. *Beamforming* de recepción también se puede realizar usando micrófonos o antenas de radar.

Con los sistemas de banda estrecha, el retardo de tiempo es equivalente a un "cambio de fase", por lo que el caso del conjunto de antenas, la fase cada una de ellas se modificó una cantidad ligeramente diferente (se dice entonces que las antenas están *en fase*). Un sistema de banda estrecha, por ejemplo radares, es aquél donde el ancho de banda es sólo una pequeña fracción de la frecuencia central. Con los sistemas de banda ancha esta aproximación ya no se sostiene, como sucede con los sónares.

En el receptor, la señal de cada antena puede ser amplificad en distinta magnitud. Diferentes patrones de ponderación (por ejemplo, *Dolph-Chebyshev*) se puede utilizar para alcanzar el nivel de sensibilidad deseado.

Las técnicas de *beamforming* se pueden dividir en dos categorías: convencionales (haz fijo) o de adaptación (antenas en fase).

Beamforming convencional utiliza un conjunto fijo de ponderaciones o pesos y retrasos de tiempo (o fases) para combinar las señales del conjunto de sensores, principalmente utilizando sólo la información sobre la ubicación de los sensores en el espacio y las direcciones de onda de interés. Por el contrario, las técnicas de formación de haces de adaptación generalmente combinan esta información con propiedades de las señales recibidas por los sensores del conjunto, por lo general para mejorar el rechazo de las señales no deseadas de otras direcciones. Este proceso puede llevarse a cabo en el dominio del tiempo y de frecuencia.

Un sistema formador de haz adaptativo es aquél en el que se realiza un procesamiento de señal seleccionando un rango espacial usando un conjunto de antenas de radar (o antenas en fase) con el fin de transmitir o recibir señales en diferentes direcciones sin tener que dirigir mecánicamente el conjunto de sensores. La distinción principal entre un

sistema de *beamforming* adaptativo y un sistema convencional es la capacidad del sistema de ajustar su rendimiento para adaptarse a cambios en su entorno.

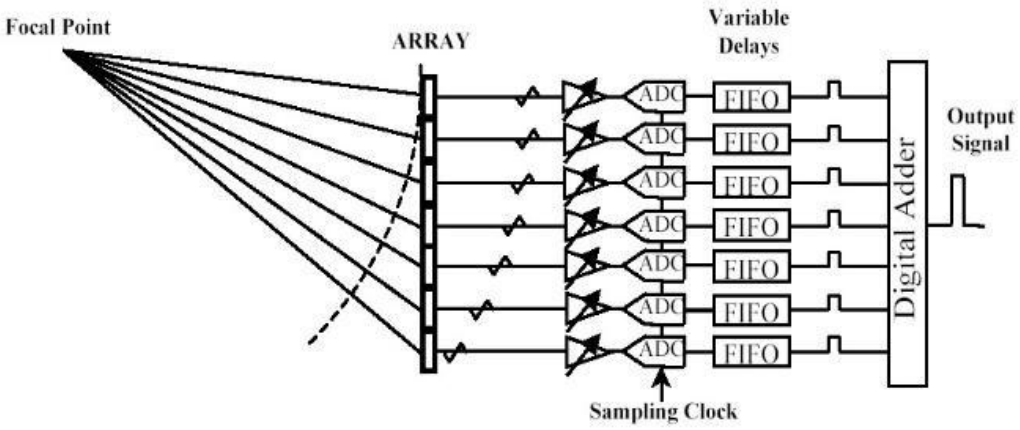


Ilustración 71 Beamforming (2)

Una característica particularmente importante en aplicaciones militares es la posibilidad de un formador de haz adaptativo de reducir la sensibilidad a ciertas direcciones de llegada para contrarrestar la interferencia de transmisiones hostiles.

Beamforming aplicado al reconocimiento de fuentes sonoras

La técnica de formación de haz o *beamforming* se puede utilizar para tratar de extraer fuentes de sonido en una habitación, como sucede en el *cocktail party problem,* donde un grupo de gente está hablando simultáneamente en una habitación (como en una fiesta de cócteles) y una persona está intentando seguir una de las conversaciones. El cerebro humando puede realizar este proceso con facilidad, pero es un problema de difícil solución usando procesamiento de señales digitales.

El problema reside en que es necesario conocer la ubicación de las fuentes de sonido de antemano, por ejemplo, utilizando la diferencia del tiempo de llegada de las fuentes a los micrófonos del array, para después calcular la ubicación a partir de las distancias.

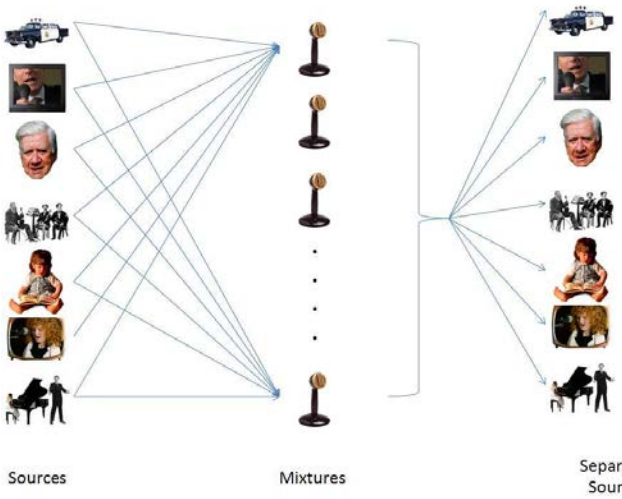


Ilustración 72 Cocktail Party Problem

Es útil usar bancos de filtros especializados para separar las bandas de frecuencia antes de la formación de haz. Esto es debido a que dependiendo de la frecuencia es necesario usar un filtro u otro para optimizar el proceso, de forma que puedan ser tratados como problemas distintos (es decir, se ejecutan múltiples filtros en paralelo, y luego se recombinan las bandas).

4.1.3. Localización de fuente sonora

Acoustic source localization, en español *Localización de fuente sonora*, es el proceso de localizar una fuente de sonido a partir de las medidas del campo acústico. El campo acústico puede ser descrito mediante magnitudes físicas como la presión del sonido o la velocidad de las partículas. Mediante la medición de estas propiedades es posible obtener una dirección de origen de forma indirecta.

Tradicionalmente, la presión del sonido se mide usando micrófonos. Los micrófonos tienen un patrón polar que describe su sensibilidad como función de la dirección del sonido incidente. Muchos micrófonos tienen un patrón polar omnidireccional que hace que su sensibilidad sea independiente de la dirección del sonido incidente. Existen también micrófonos con otros patrones polares más sensibles en una dirección determinada.

Además de considerar micrófonos que miden la presión del sonido, también es posible utilizar una sonda para medir la velocidad de las partículas sonoras directamente. La velocidad de las partículas es otra magnitud relacionada con las ondas sonoras pero, a diferencia de la presión de sonido, la velocidad de las partículas es un vector. Mediante la medición de la velocidad de las partículas se obtiene una dirección de la fuente directamente.

Otros métodos para localizar espacialmente una fuente sonora son más complicados y utilizan múltiples sensores. Muchos de estos métodos utilizan la técnica de diferencia de tiempo de llegada (*Time Difference Of Arrival*).

Algunos han denominado el proceso de localización de la fuente acústica como un problema de ingeniería inversa donde el campo de sonido medido se traduce a la posición de la fuente de sonido.

Existen diferentes métodos usados en la actualidad para resolver este problema. Algunos de ellos son:

- *Velocidad de partículas sonoras o vector de intensidad*
  Es la forma más simple y es relativamente nueva. Se basa en medir la velocidad de las partículas sonoras utilizando una sonda de velocidad de las partículas. La velocidad de las partículas es un vector y por tanto también contiene información direccional.

- *Time difference of arrival (TDOA)*
  Es la forma más simple y es relativamente nueva. Se basa en medir la velocidad de las partículas sonoras utilizando una sonda de velocidad de las partículas. La velocidad de las partículas es un vector y por tanto también contiene información direccional.

  Con un array de sensores (por ejemplo un array de micrófonos) que conste de al menos dos sensores es posible obtener la dirección de la fuente utilizando la función de correlación cruzada entre la señal de cada uno de los sensores. La función de correlación cruzada entre dos micrófonos se define como:

$$R_{x_1,x_2}(\tau) = \sum_{n=-\infty}^{\infty} x_1(n)\, x_2(n+\tau)$$

  Esta función define el nivel de correlación entre las salidas de dos sensores x_1 y x_2. En general, un mayor nivel de correlación significa que el argumento τ es relativamente cercano al valor real de TDOA. Para dos sensores uno junto al otro el valor de TDOA viene dado por:

$$\tau_{\mathrm{true}} = \frac{d_{\mathrm{spacing}}}{c}$$

Donde c es la velocidad del sonido en el medio que rodea los sensores y la fuente. Un ejemplo conocido de TDOA es la diferencia de tiempo interaural (Interaural Time Difference), magnitud que mide para animales y humanos la diferencia de tiempo en la llegada de un sonido entre los dos oídos. Viene dada por:

$$\Delta t = \frac{x \sin \theta}{c}$$

Donde:

Δt es la diferencia de tiempo en segundos.

x es la distancia entre los dos sensores (oídos) en metros.

θ es el ángulo entre la base de los sensores (oídos) y el sonido incidente en grados.

- *Triangulación*

  En trigonometría y geometría, triangulación es el proceso de determinar la localización de un punto midiendo ángulos hasta el punto en cuestión desde otros puntos conocidos usando los extremos de una línea base, en vez de medir distancias al punto directamente. El punto puede entonces ser fijado como un tercer punto de un triángulo con un lado y dos ángulos conocidos.

  Para el proceso de localización de sonido, esto significa que si la dirección de la fuente puede ser medida desde al menos dos puntos en el espacio, es posible calcular su posición.

4.2. Análisis de sensibilidad

Como se ha visto en anteriores capítulos, el sensor Kinect es capaz de proporcionar con un grado de confianza el valor del ángulo calculado por algoritmos de *beamforming* dada una fuente acústica.

A diferencia de la detallada documentación existente respecto a los rangos de operación del sensor Kinect para la obtención de datos de imagen de profundidad (representado en la Ilustración 74), Microsoft no proporciona unos rangos de operación específicos para las funcionalidades de audio de Kinect. Es decir, las especificaciones del sensor únicamente indican que el rango de operación de la funcionalidad de localización de fuente sonora es de 50 grados a cada lado del sensor, pero no se indican más detalles respecto a las distancias de operación u otras variables que puedan influir.

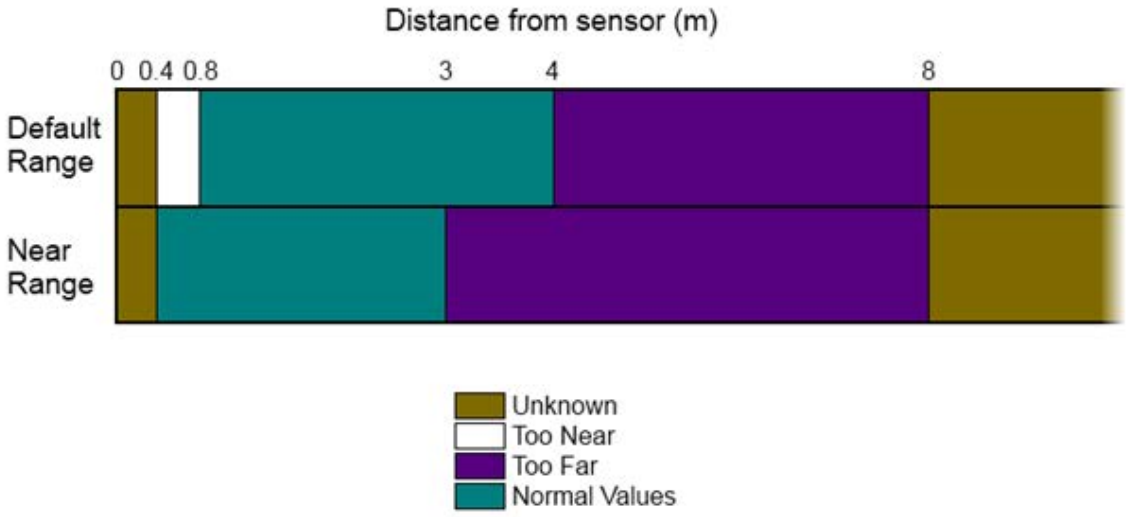


Ilustración 73 Rango de operación del sensor de profundidad de Kinect

Es objetivo del presente análisis, por tanto, resolver las siguientes preguntas:

- ¿Cuáles son los rangos de operación de Kinect para la recepción de audio?
- ¿Existe una distancia mínima o máxima para la cual no es óptimo usar la funcionalidad de *beamforming*? En general, ¿Cómo afecta la distancia al sensor para la precisión en el cálculo del ángulo de origen?
- ¿Afecta el escenario de uso (espacios cerrados, abiertos, tamaño de sala, etc.)?

Estas tres preguntas sin respuesta son el motivo por el que se haya decidido realizar un estudio sobre la sensibilidad del dispositivo Kinect en distintos escenarios y con el objetivo de analizar el grado de confianza que se puede aplicar a la precisión del localizador del ángulo de origen.

Por ello se realizarán tres pruebas que analizan las variables distancia y escenario:

- Espacio cerrado pequeño, distancias entre 1 y 3 metros.
- Espacio cerrado amplio, distancias entre 1 y 8 metros.
- Espacio abierto al aire libre, distancias entre 1 y 8 metros.

Para cada prueba se documentará meticulosamente la descripción del escenario usado, se representarán los resultados obtenidos por medio de gráficos representativos y se realizarán conclusiones para cada prueba.

El procedimiento a seguir para cada prueba es el mismo:

- Se coloca el sensor Kinect en el lateral del escenario que permita mayor movilidad para realizar la prueba.

- Se calcula desde la frontal del sensor el rango de recepción de audio, 100 grados, lo que supone 50 a cada lado. Para ello se traza primero una línea recta desde el sensor y con ayuda de un transportador de ángulos se trazan las direcciones de los ángulos +50 grados y -50 grados desde la frontal, siendo los valores negativos los localizados a la derecha del sensor desde su punto de vista y a la izquierda de la persona que interacciona.
- Cada sección de 50 grados se divide en 4 de forma que se pueda analizar con más detalle la precisión de Kinect. Se representan, por tanto, los ángulos -50, -37,5, -25, -12,5, 0, +12,5, +25, +37,5 y +50 grados. En la Ilustración 75 se muestra esta división.
- A continuación, dependiendo de la prueba, se marcarán en el escenario las distancias a las que se vaya a probar el sensor. Estas distancias medidas en metros desde el sensor, definirán junto con los citados ángulos, las posiciones en las que se realizará la prueba. Cada distancia se mide desde el sensor para cada dirección, formando arcos.

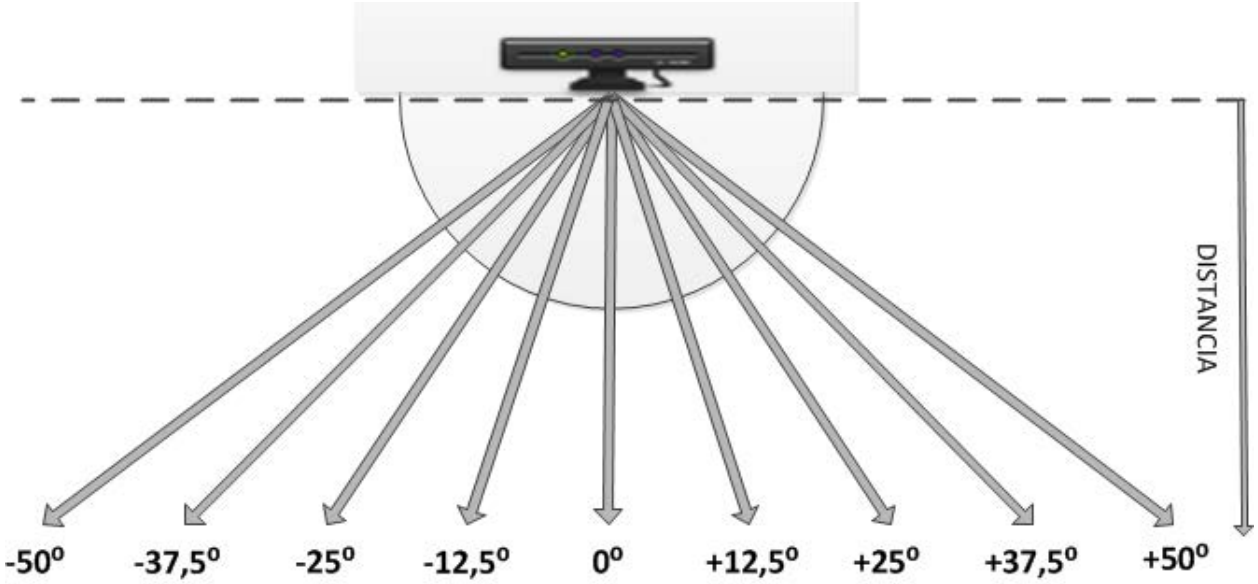


Ilustración 74 Configuración general de la prueba

- Se hará uso del programa de referencia Kinect Audio Demo incluido en el SDK de Kinect para Windows 1.0 que se describió anteriormente, y se mantendrá en ejecución en un equipo portátil conectado al sensor Kinect. El ejecutor de la prueba se desplazará a cada punto marcado en el escenario.
- En cada punto de prueba, se reproducirá la misma grabación desde un dispositivo móvil (HTC Desire S). En dicha grabación se almacena la frase **"El veloz murciélago hindú comía feliz cardillo y kiwi"**. Esta frase se repite tres veces. En total, la grabación dura aproximadamente 7 segundos.
- El programa Kinect Audio Demo muestra el ángulo y grado de confianza que Kinect calcula para esta señal acústica de entrada y la representa gráficamente con una rotación de un medidor. En la Ilustración 76 se muestra un ejemplo de su funcionamiento.

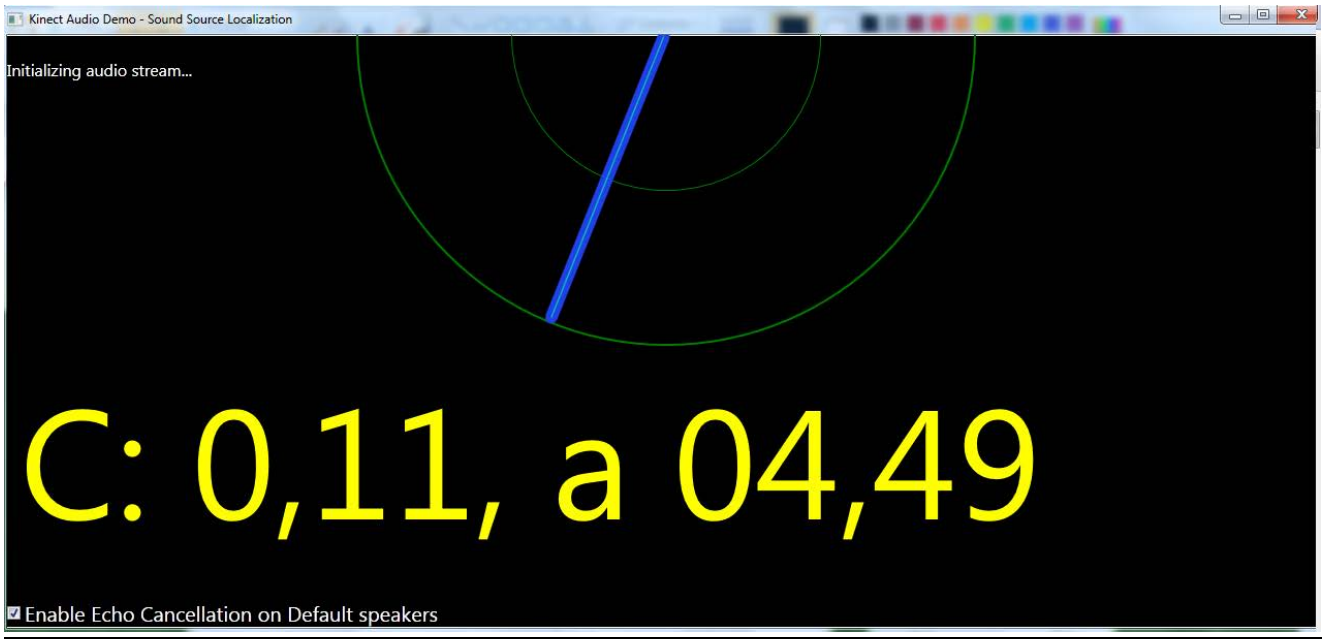


Ilustración 75 Kinect Audio Demo

El primer valor mostrado es el grado de confianza "**C**", mientras que el segundo "**a**" muestra el ángulo calculado por el sensor. En el ejemplo de la Ilustración PEPE, el ángulo calculado es 4,5 grados, por lo que el indicador ha rotado esa cantidad.

- El analizador de la prueba registra el valor mostrado así como el grado de confianza para cada punto de la prueba.
- Finalmente, se analiza el resultado y se obtienen conclusiones.

4.2.1. Prueba 1: Espacio cerrado pequeño

Descripción del escenario

La primera de las pruebas a realizar tiene como escenario una habitación de dormitorio con las siguientes características:

- Dimensiones:
  - Ancho: 4 metros
  - Largo: 4 metros
  - Alto: 3 metros
- Decorado: La habitación se encuentra amueblada con un armario empotrado, una estantería, mesa de trabajo, equipo informático y una cama.

Se ha escogido este escenario para realizar la prueba por dos razones:

- Sus dimensiones la hacen idónea para una prueba del sensor Kinect en distancias cortas (hasta 3 metros).
- Su decorado impide que se produzca efecto eco.

A continuación se muestran dos imágenes del escenario.

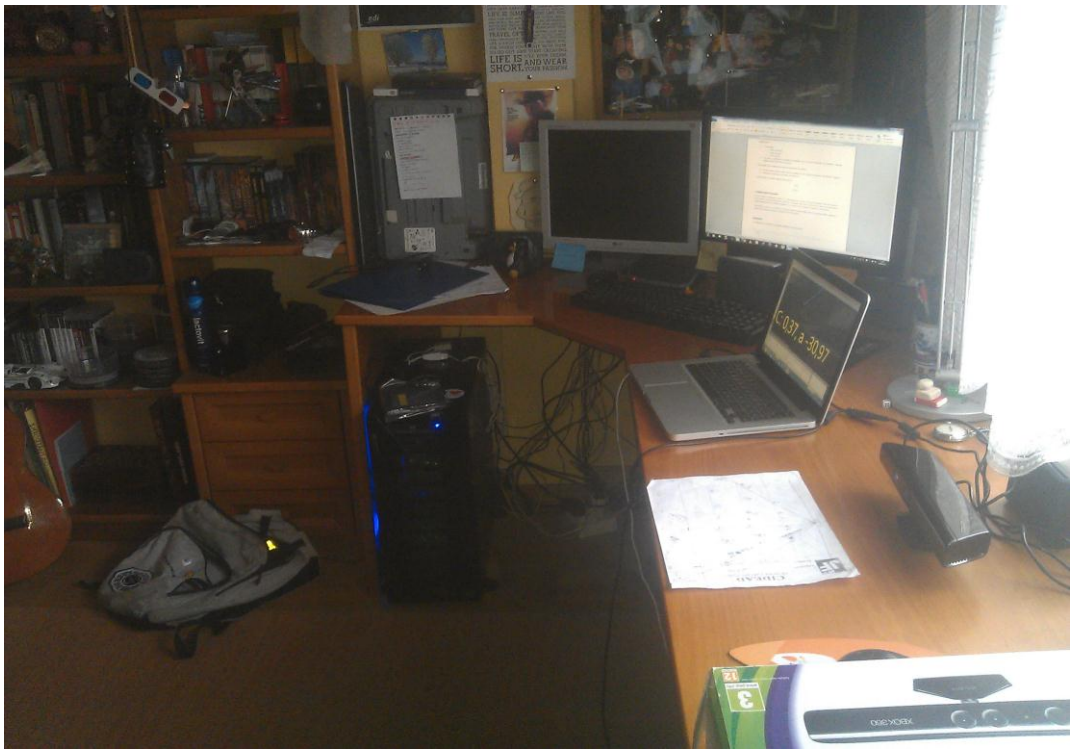


Ilustración 76 Escenario de la Prueba 1 (1)

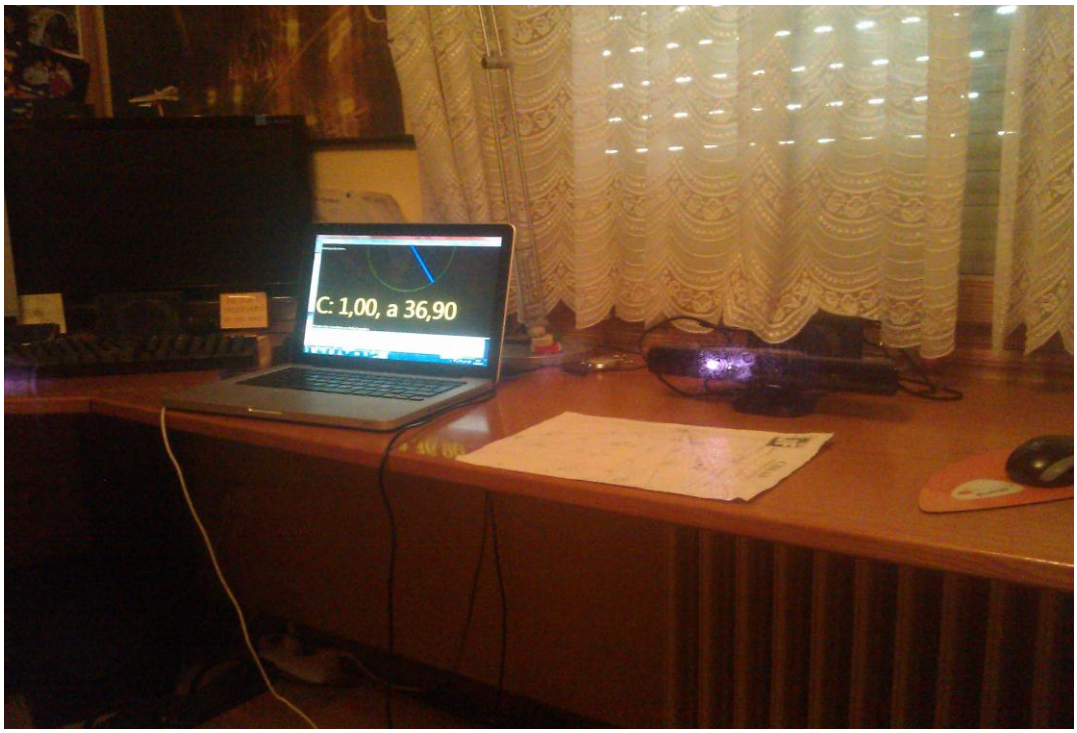


Ilustración 77 Escenario de la Prueba 1 (2)

Configuración de la prueba

En esta prueba se va a probar el sensor en corta distancia, por lo que se marcará el escenario cada metro desde 1 hasta 3 metros para cada dirección del rango [-50, 50] grados según se estableció antes. Esto forma un total de 9 puntos de prueba para la misma distancia, siendo las distancias posibles 1, 2 y 3 metros, hay un total de 27 puntos en esta prueba.

Cada punto se marca con un indicador, se inicia la aplicación Kinect Audio Demo con el sensor Kinect conectado a la toma de corriente y al equipo, y se procede a iniciar la prueba.

Resultados

A continuación se muestran los resultados obtenidos para esta prueba en distintos formatos.

Distancia / Ángulo	-50°	-38°	-25°	-13°	0°	12,5°	25°	37,5°	50°
1 metro	-45	-35	-30	-10	5	11	20	40	47
2 metros	-46	-35	-31	-14	6	11	20	42	47
3 metros	-47	-35	-30	-12	5	12	20	43	48

Tabla 3 Resultados Prueba 1

Los valores en sombreado amarillo de la Tabla 3 representan los valores de los ángulos reales sobre los que se realiza la medición. Los azules, la distancia. A continuación se muestra la diferencia de entre estos valores:

Distancia / Diferencia	-50°	-37,5°	-25°	-12,5°	0°	12,5°	25°	37,5°	50°
1 metro	5	2,5	5	2,5	5	1,5	5	2,5	3
2 metros	4	2,5	6	1,5	6	1,5	5	4,5	3
3 metros	3	2,5	5	0,5	5	0,5	5	5,5	2

Tabla 4 Diferencia entre valores reales y calculados. Prueba 1

El valor del grado de confianza que se registraron es prácticamente idéntico para la misma distancia. Sus valores se muestran a continuación.

Distancia	1 metro	2 metros	3 metros
Grado de confianza	0.5-0.6	0.6-0.8	0.8-1.0

Ilustración 78 Resultados de grado de confianza. Prueba 1

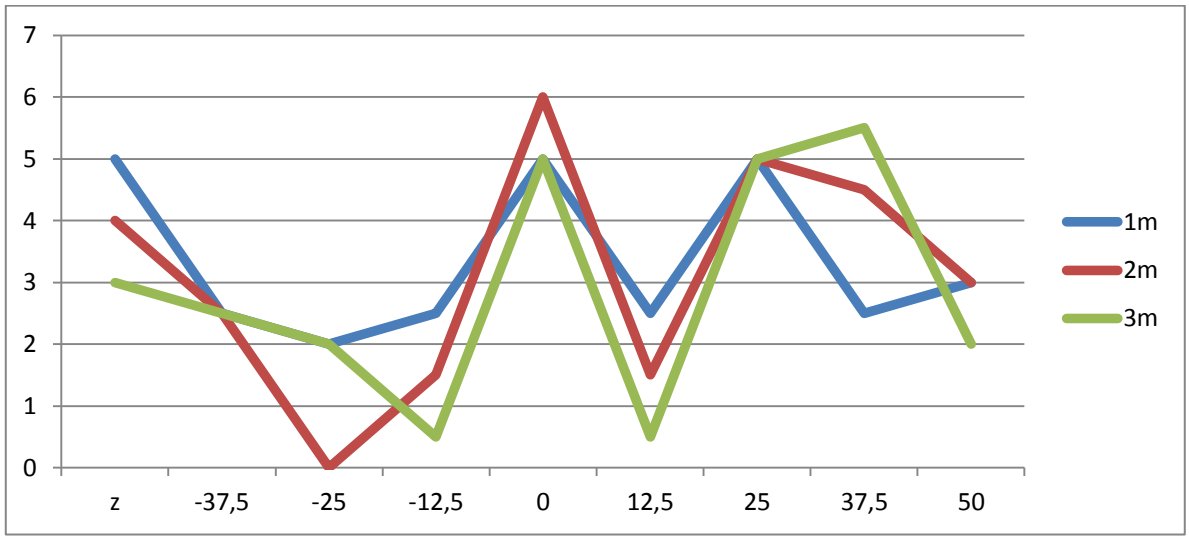


Ilustración 79 Representación gráfica de la diferencia de valores. Prueba 1

Conclusiones

Tras finalizar la prueba, y dados los resultados obtenidos, se observan las siguientes conclusiones:

- Existe un desvío entre el valor real y calculado del ángulo de origen de sonido. Este valor se manifiesta con mayor evidencia en las direcciones de 0, +25 y -25 grados, donde el sensor llega a marcar una diferencia de hasta 6 grados. La media de esta diferencia es de **3 grados**.
- Se observa una mínima pérdida de precisión conforme la distancia aumenta (2 grados aproximadamente). Se esperará a posteriores pruebas para obtener conclusiones respecto a este punto.
- El grado de confianza claramente mejora con la distancia.

4.2.2. Prueba 2: Espacio cerrado amplio

Descripción del escenario

Esta prueba se sitúa en un entorno cerrado y amplio. Sus características son:

- Dimensiones:
  - Ancho: 4 metros
  - Largo: 8 metros
  - Alto: 2,5 metros
- Decorado: Se trata de un garaje, en concreto la habitación destinada a alojar el vehículo. Dispone de dos ventanas, una puerta principal y otras dos puertas hacia habitaciones más pequeñas.

Se ha escogido este escenario para realizar la prueba por dos razones:

- Sus dimensiones la hacen idónea para una prueba del sensor Kinect en distancias cortas (hasta 3 metros) en un entorno cerrado.
- Su decorado (prácticamente inexistente) hace que se produzca efecto eco, aspecto que se desea probar para comprobar si afecta a la precisión del sensor (la puerta principal se cierra para la prueba).

A continuación se muestra una imagen del escenario.

Ilustración 80 Escenario de la Prueba 2

Resultados

Ya que en este escenario entra en juego el efecto eco, y teniendo en cuenta que el API de Kinect ofrece funcionalidades de cancelación de eco, se va a realizar esta prueba por duplicado. Primero, con la funcionalidad de cancelación de eco desactivada y posteriormente, activada.

Sin cancelación de eco

A continuación se muestran los resultados obtenidos para esta prueba en distintos formatos.

Distancia / Ángulo	-50°	-37,5°	-25°	-12,5°	0°	12,5°	25°	37,5°	50°
1 metro	-45	-32	-16	-5	7	20	28	35	48
2 metros	-43	-33	-20	-5	7	17	28	32	47
3 metros	-40	-30	-20	-5	7	17	29	31	45

Tabla 5 Resultados Prueba 2 (sin cancelación de eco)

Los valores en sombreado amarillo de la Tabla 5 representan los valores de los ángulos reales sobre los que se realiza la medición. Los azules, la distancia. A continuación se muestra la diferencia de entre estos valores:

Distancia / Diferencia	-50°	-37,5°	-25°	-12,5°	0°	12,5°	25°	37,5°	50°
1 metro	5	5,5	9	7,5	7	7,5	3	2,5	2
2 metros	7	4,5	5	7,5	7	4,5	3	5,5	3
3 metros	10	7,5	5	7,5	7	4,5	4	6,5	5

Tabla 6 Diferencia entre valores reales y calculados. Prueba 2 (sin cancelación de eco)

El valor del grado de confianza registrado para esta prueba difiera respecto al registrado en la primera prueba.

Grado de confianza	-50	-37,5	-25	-12,5	0	12,5	25	37,5	50
1 metro	0.6	0.5	1.0	1.0	1.0	1.0	0.6	0.5	0.5
2 metros	0.4	0.4	1.0	1.0	1.0	0.6	1.0	0.6	0.4
3 metros	0.3	0.4	0.5	0.6	1.0	0.8	0.6	0.4	0.3

Ilustración 81 Resultados de grado de confianza. Prueba 2 (sin cancelación de eco)

Mientras que en la primera prueba su valor era más o menos constante a misma distancia, en esta prueba se observa una pérdida de precisión conforme los registros se alejan en distancia y en ángulos más alejados.

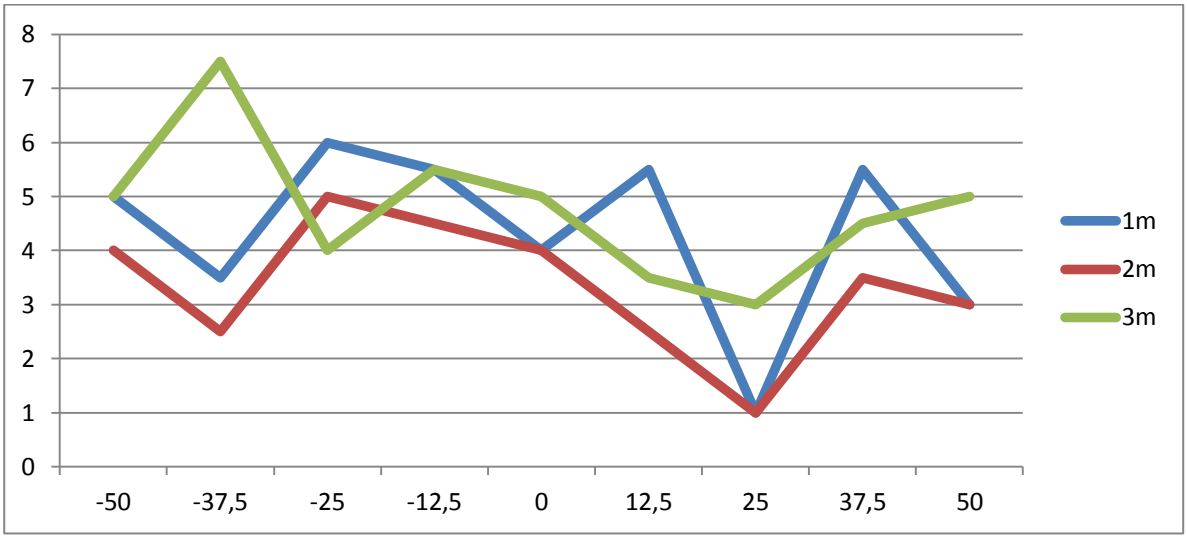


Tabla 7 Representación gráfica de la diferencia de valores. Prueba 2 (sin cancelación de eco)

Conclusiones

- Se comprueba que el grado de confianza se corresponde con la diferencia entre el valor del ángulo real y el calculado: A menor grado de confianza, mayor es la diferencia. Este resultado es lógico puesto que el grado de confianza devuelto por el sensor mide la precisión con la que el sensor calcula el ángulo.
- El eco afecta a la precisión del cálculo. En este caso, la diferencia entre valor real y calculado asciende en algunos casos hasta 10 grados, siendo de media de **6 grados.**

Con cancelación de eco

Para realizar este caso particular se habilita en el ejemplo Kinect Audio Demo la opción “Enable Echo Cancellation on Default Speakers”.

Ilustración 82 Opción de activación de eco de Kinect Audio Demo

Distancia / Ángulo	-50°	-37,5°	-25°	-12,5°	0°	12,5°	25°	37,5°	50°
1 metro	-45	-34	-19	-7	6	18	26	32	47
2 metros	-46	-35	-20	-8	5	15	26	34	47
3 metros	-45	-30	-21	-7	5	16	28	33	45

Tabla 8 Resultados Prueba 2 (con cancelación de eco)

Los valores en sombreado amarillo de la Tabla 8 representan los valores de los ángulos reales sobre los que se realiza la medición. Los azules, la distancia. A continuación se muestra la diferencia de entre estos valores:

Distancia / Diferencia	-50°	-37,5°	-25°	-12,5°	0°	12,5°	25°	37,5°	50°
1 metro	5	3,5	6	5,5	4	5,5	1	5,5	3
2 metros	4	2,5	5	4,5	4	2,5	1	3,5	3
3 metros	5	7,5	4	5,5	5	3,5	3	4,5	5

Tabla 9 Diferencia entre valores reales y calculados. Prueba 2 (con cancelación de eco)

El valor del grado de confianza registrado para esta prueba difiera respecto al registrado en la primera prueba.

Grado de confianza	-50	-37,5	-25	-12,5	0	12,5	25	37,5	50
1 metro	0.6	0.7	1.0	1.0	1.0	1.0	0.7	0.7	0.6
2 metros	0.6	0.7	1.0	0.9	1.0	0.6	1.0	0.6	0.8
3 metros	0.6	0.7	0.6	0.8	1.0	0.9	0.7	0.5	0.6

Tabla 10 Resultados de grado de confianza. Prueba 2 (con eco)

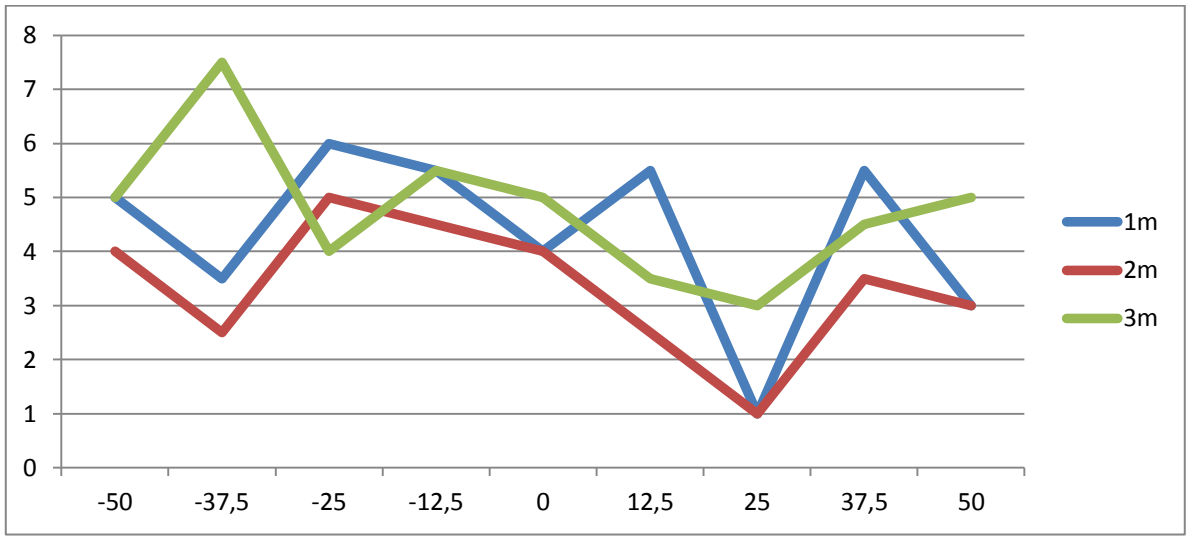


Ilustración 83 Representación gráfica de la diferencia de valores. Prueba 2 (con cancelación de eco)

Conclusiones

- Al activar la funcionalidad de cancelación de eco, los resultados hablan por sí solos: La diferencia media entre los valores reales y obtenidos por el sensor ha variado de **6 grados** sin usar cancelación de eco a **4 grados**.
- El grado de confianza ha mejorado sustancialmente, siendo el sensor más preciso en el valor devuelto.

4.2.3. Prueba 3: Espacio abierto

Descripción del escenario

Para esta última prueba se ha escogido como escenario un jardín doméstico. Sus características son:

- Dimensiones: superficie adecuada para abarcar el rango de hasta 8 metros de la prueba.
- Decorado: la zona de pruebas se encuentra despejada de plantas y demás vegetación. Sin embargo, existe una mesa con sillas en la sección situada entre el rango de visión del sensor en el rango [-50, 0] grados.

Se ha escogido este escenario para realizar la prueba por dos razones:

- Sus dimensiones la hacen idónea para una prueba del sensor Kinect en distancias medias (hasta 8 metros).
- ¿La existencia de otros elementos en el escenario influye en la precisión del sensor?

A continuación se muestra una imagen del escenario.

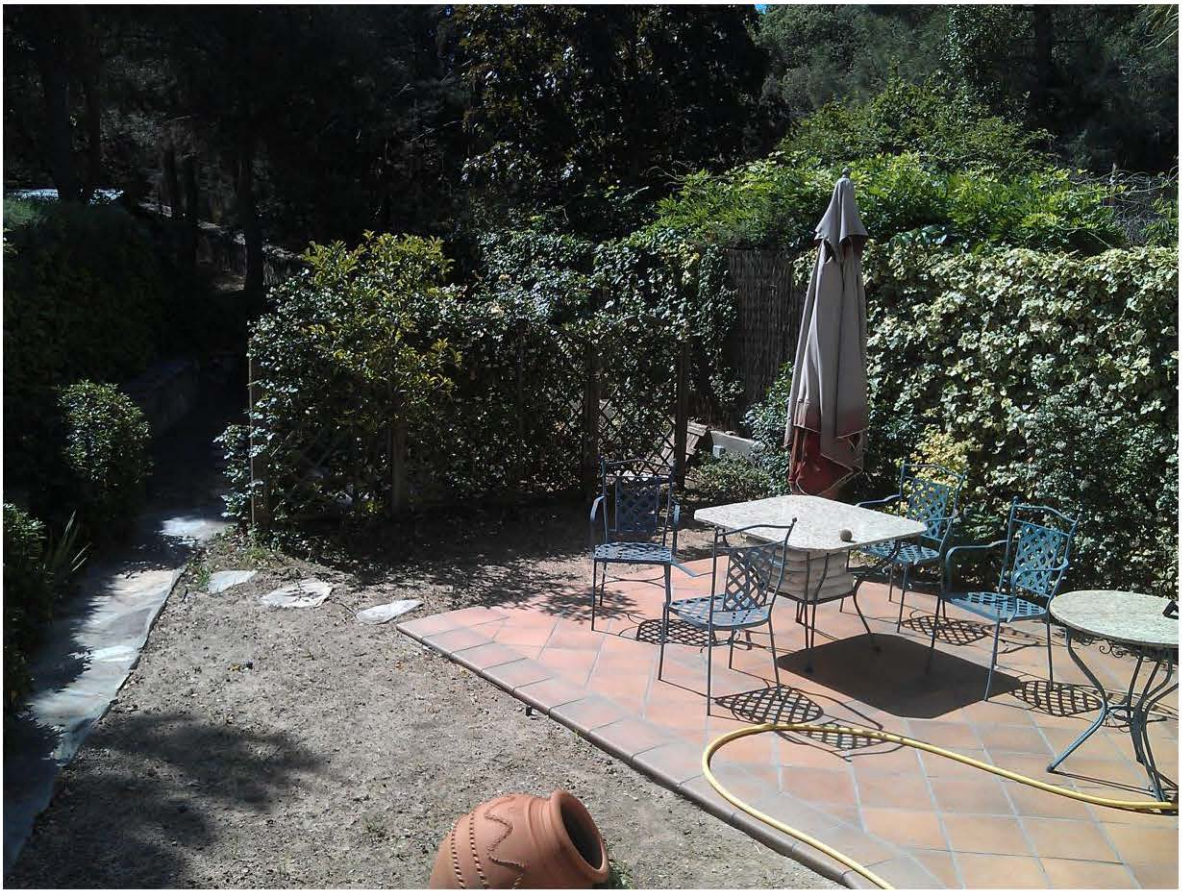


Tabla 11 Escenario. Prueba 3

Resultados

A continuación se muestran los resultados obtenidos para esta prueba en distintos formatos.

Distancia/ Ángulo	-50	-37,5	-25	-12,5	0	12,5	25	37,5	50
1 metro	-40	-29	-10	-4	7	17	30	40	48
2 metros	-39	-29	-12	-4	7	17	30	40	45
3 metros	-40	-30	-13	-5	7	17	30	42	48
4 metros	-40	-28	-12	-5	7	17	32	42	48
5 metros	-41	-28	-13	-5	7	17	30	43	48
6 metros	-41	-28	-13	-5	7	17	30	42	48
7 metros	-41	-28	-13	-5	7	17	30	41	48
8 metros	-41	-28	-13	-5	7	17	32	45	48
9 metros	-41	-29	-13	-5	7	17	30	45	48

Tabla 12 Resultados Prueba 3

Diferencia / Distancia	-50	-37,5	-25	-12,5	0	12,5	25	37,5	50
1 metro	10	8,5	15	8,5	7	4,5	5	2,5	2
2 metros	11	8,5	13	8,5	7	4,5	5	2,5	5
3 metros	10	7,5	12	7,5	7	4,5	5	4,5	2
4 metros	10	9,5	13	7,5	7	4,5	7	4,5	2
5 metros	9	9,5	12	7,5	7	4,5	5	5,5	2
6 metros	9	9,5	12	7,5	7	4,5	5	4,5	2
7 metros	9	9,5	12	7,5	7	4,5	5	3,5	2
8 metros	9	9,5	12	7,5	7	4,5	7	7,5	2
9 metros	9	8,5	12	7,5	7	4,5	5	7,5	2

Tabla 13 Diferencia entre valores reales y calculados. Prueba 3

Grado de confianza	-50	-37,5	-25	-12,5	0	12,5	25	37,5	50
1 metro	1	1	0.6	1	1	1	0.7	0.7	1
2 metros	1	1	0.7	1	1	1	0.6	0.6	1
3 metros	1	1	0.7	1	1	1	1	0.5	1
4 metros	1	1	0.7	1	1	1	0.6	0.5	1
5 metros	1	1	0.5	1	1	1	0.6	0.6	1

6 metros	1	1	0.6	1	1	1	0.7	0.6	1
7 metros	1	1	0.6	1	1	1	0.7	0.6	1
8 metros	1	1	0.6	1	1	1	0.7	0.7	1
9 metros	1	1	0.6	1	1	1	0.7	0.6	1

Tabla 14 Resultados de grado de confianza. Prueba 3

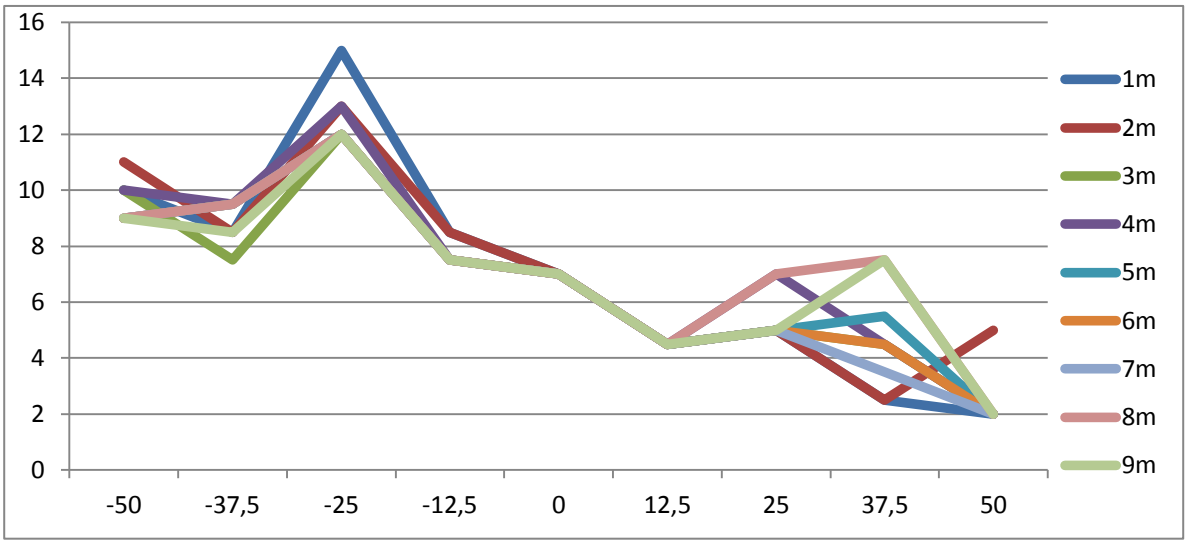


Tabla 15 Representación gráfica de la diferencia de valores. Prueba 3

Conclusiones

Esta es la prueba realizada en el ambiente más amplio, y dados los resultados obtenidos, donde se encuentran los mayores contrastes:

- Mientras que los valores de la diferencia mantienen la media de **6 grados** en la zona de recepción situada a la izquierda desde el punto de vista de Kinect, la zona situada a la derecha muestra un desnivel de hasta **14 grados**. Esto se explica por la existencia de obstáculos en esa zona, por lo que se concluye que el sensor es menos preciso si entre la fuente sonora y el sensor se encuentran obstáculos. En concreto, se observa cómo justo en el lugar donde se encontraba el obstáculo, aproximadamente en la dirección -25 grados, la precisión muestra su valor más bajo.
- El grado de confianza es el más constante de las 3 pruebas realizadas. Podría atribuirse a la ausencia completa de efecto eco.

4.2.4. Conclusiones generales

Para finalizar este apartado, se muestran las conclusiones a las que se han llegado a nivel general en vista de los resultados obtenidos. Respondiendo a las preguntas propuestas al inicio del presente apartado:

- ¿Cuáles son los rangos de operación de Kinect para la recepción de audio?
  En entornos pequeños, los resultados sugieren que la distancia mínima sería de **1 metro.** Como rango máximo no se ha encontrado un valor a partir del cual la precisión caiga drásticamente, por lo que no se puede especificar. Por tanto el sensor podría ser utilizado con toda probabilidad **a más de 8 metros** con el fin de localizar una fuente sonora.

- ¿Existe una distancia mínima o máxima para la cual no es óptimo usar la funcionalidad de *beamforming*?
  No se ha encontrado una pérdida de precisión suficiente como para poder afirmar que existen rangos máximos. **En general, el sensor Kinect pierde precisión según aumenta la distancia entre la fuente sonora y el sensor.** En concreto, **este resultado es más manifiesto en entornos donde se manifieste efecto eco.**

- En general, ¿Cómo afecta la distancia al sensor para la precisión en el cálculo del ángulo de origen?
  La pérdida de precisión conforme la distancia al sensor es ínfima (del grado de **5 grados** como máximo). Por tanto, la respuesta es que la distancia afecta, como no puede ser de otra forma al tratarse de un conjunto de micrófonos, pero a un **nivel despreciable**. Son las funcionalidades de cancelación de eco o ajuste automático de ganancia las que hacen que el factor distancia no suponga un problema para la estabilidad del sensor Kinect.

- ¿Afecta el escenario de uso (espacios cerrados, abiertos, tamaño de sala, etc.)?
  La respuesta es sí. En espacios cerrados el sensor es más estable en cuanto a los resultados que en espacios amplios. Se ha comprobado que la existencia de obstáculos entre la fuente acústica y el sensor produce una pérdida de precisión.

Se ha demostrado por tanto que el sensor Kinect es un dispositivo de alta calidad, a pesar de que los valores calculados para la localización de audio nunca se acerquen aproximadamente a menos de **2 grados** del valor real.

En concreto, esta desviación es más o menos constante a 6 o 7 grados **más** entre el rango -50 a +12,5 grados.

Es decir, el sensor calcula el valor con un desvío general de -5 grados:

- Si el valor real es -25 grados, devuelve entre -16 y -20 grados.
- Si el valor real es 12,5 grados, devuelve entre 17 y 20 grados

Por otro lado, si el ángulo real es superior a 25 grados, el sensor comienza a devolver valores más cercanos al real (con más precisión) pasando a devolver valores con un desvío **por debajo del real:**

- Si el valor real es +25 grados, el desvío general es de 2 o 3 grados por encima (27-29 grados).
- Si el valor real es +37,5 grados, el valor devuelto es **inferior**: 32-36 grados.
- Si el valor real es +50 grados, el valor devuelto es +47 o +48 grados.

Por tanto, **se observa un desvío constante entre -50 y 25 grados de aproximadamente +6 grados, siendo el valor más preciso el de +37,5 grados, y terminando con un desvío de -3 grados para el valor real de +50 grados.**

Como conclusión, se recomienda el uso de este sensor para labores de grabación de audio, localización de fuente acústica y cualquier otra aplicación práctica que pueda necesitarse de una matriz de micrófonos. Su estabilidad frente

a situaciones difíciles de manejar desde el punto de vista de algoritmia y limpieza de señales como la existencia de ruidos residuales (altavoces cercanos) o efecto eco hacen de este sensor un producto de gran calidad.

5. Capítulo 5: Prueba de concepto

En el capítulo 2 se presentó el sensor Kinect y se prestó especial atención a los usos en que la comunidad le ha dado aplicado a distintos ámbitos: medicina, robótica, animación, tecnología, marketing, ocio, etc. Todas estas aplicaciones se han basado en mayor o menor medida en la cámara de profundidad del sensor Kinect, y pocas o ninguna usa siquiera las funcionalidades de audio del sensor. No es de extrañar, puesto que el sensor de profundidad es sin duda su componente más interesante y del que más partido se puede sacar. Viejos problemas son ahora económicamente viables y una innumerable cantidad de nuevos proyectos son posibles. Podemos afirmar por tanto que el conjunto de micrófonos de Kinect ha quedado eclipsado por las funcionalidades de sus cámaras.

Dada esta situación, se ha decidido ayudar a equilibrar un poco la balanza. Con nuestra humilde contribución, vamos a mostrar un ejemplo práctico de una aplicación basada en Kinect que use las funcionalidades de tratamiento de audio. Su objetivo es por tanto demostrar que la calidad del array de micrófonos puede ser usada en aplicaciones fiables.

En el capítulo 4 hemos comprobado mediante análisis de sensibilidad la calidad del array de micrófonos del sensor. La conclusión a que se llegó es que está preparado para usarse en aplicaciones de carácter general y en las que requieran precisión y calidad en la localización de una señal acústica.

5.1. Introducción

Esta prueba de concepto tiene como objetivo servir de ejemplo del abanico de posibilidades que existen a la hora de desarrollar aplicaciones con Kinect. En concreto, los objetivos de la prueba de concepto son:

- El uso principal de la funcionalidad de localización de fuente acústica del sensor Kinect.
- El uso del conjunto de micrófonos del sensor Kinect como dispositivo de grabación de audio.
- El uso de las funcionalidades de captura de imagen.
- La obtención de imágenes en color usando como fuente una cámara PTZ.
- La creación de un fichero de vídeo usando Kinect y la cámara PTZ.
- Firma digital del vídeo usando certificado digital.

Para ello se desarrollará un programa de escritorio para Windows usando el SDK presentado en el apartado anterior. La funcionalidad principal de la aplicación es la de generar una video acta. Una video acta es una grabación de video en la que varios interlocutores mantienen una conversación y es preciso registrar un informe verídico de esa conversación. Es por tanto un medio para redistribuir los temas tratados.

Para poder reconocer qué persona está hablando en cada momento y de esa forma poder enfocar la cámara de video en su dirección se usará la funcionalidad de localización de fuente acústica de Kinect. Además de Kinect se obtendrá también el flujo de datos de audio para grabar la conversación. El resto de elementos se describen en el siguiente apartado.

5.2. Elementos del sistema

La aplicación estará formada por los siguientes elementos:

- **Aplicación cliente**: Escrita en lenguaje C# usando el SDK de Kinect para Windows versión 1.5, esta aplicación se ejecuta sobre la máquina conectada físicamente con el sensor Kinect. Mostrará los datos suministrados por la cámara RGB de Kinect, una representación del cálculo de la localización de fuente sonora y las imágenes devueltas por la cámara PTZ. A lo largo del presente apartado se describe detalladamente esta aplicación.

- **Servidor**: Se usará un programa diseñado previamente por Álvaro Luis Bustamante como soporte para esta prueba de concepto. Tendrá acceso físico a la cámara PTZ y renviará tramas de imágenes a color a la aplicación cliente.

- **Cámara PTZ**: Esta cámara se encarga de capturar las imágenes del escenario y de enviarlas al programa servidor.

- **Sensor Kinect**: El sensor sobre el que se centra este PFC. Proporcionará a la aplicación cliente de flujos de datos de imágenes a color, de audio y de seguimiento de esqueleto.

5.3. Arquitectura

En la siguiente ilustración se muestra el diagrama de arquitectura del sistema. Se trata de una arquitectura sencilla de cliente-servidor en el que la aplicación cliente conectada al sensor Kinect está conectada a la misma red que un equipo sobre el que se ejecuta la aplicación servidor. A su vez, el servidor se conecta físicamente a la cámara PTZ, de la que obtiene imágenes que retransmite a la aplicación cliente.

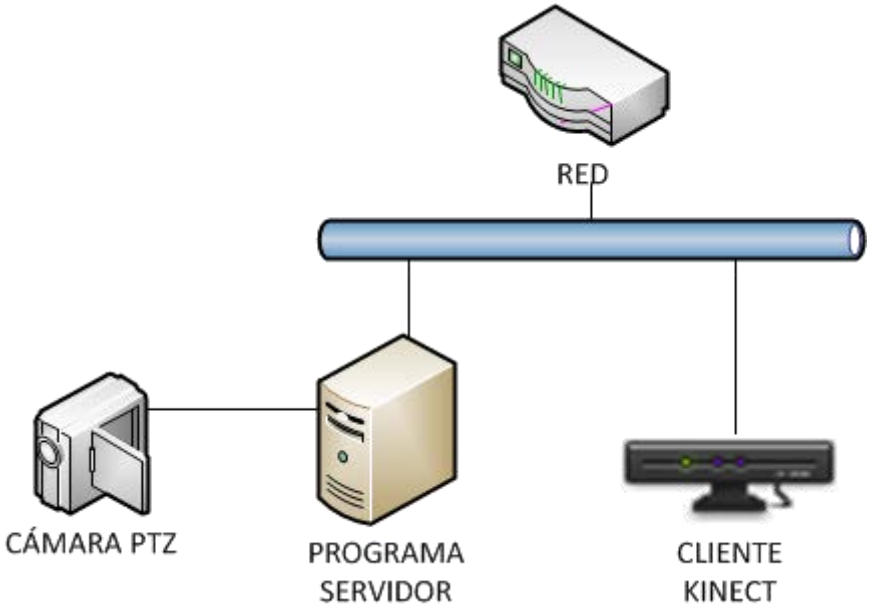


Ilustración 84 Diagrama de arquitectura del sistema

Todos los elementos de la arquitectura estarán instalados en el laboratorio del GIAA. La red a usar será la que actualmente usa el laboratorio como red interna cableada.

5.4. Escenario

El sistema se ha desarrollado y probado en el laboratorio del GIAA del Campus de Colmenarejo de la Universidad Carlos III de Madrid.

La elección de este escenario se debe a que las cámaras PTZ están instaladas en el laboratorio y son fijas. Esto, sumado a que la conexión a la red interna y por tanto al servidor que controla la cámara es posible únicamente mediante conexión Ethernet cableada, hace que este sea el único escenario posible para la prueba de concepto.

En las ilustraciones 86 y 87 se puede observar el laboratorio. Se trata de una habitación de trabajo con dimensiones aproximadas de 7 metros de largo por 5 metros de ancho.

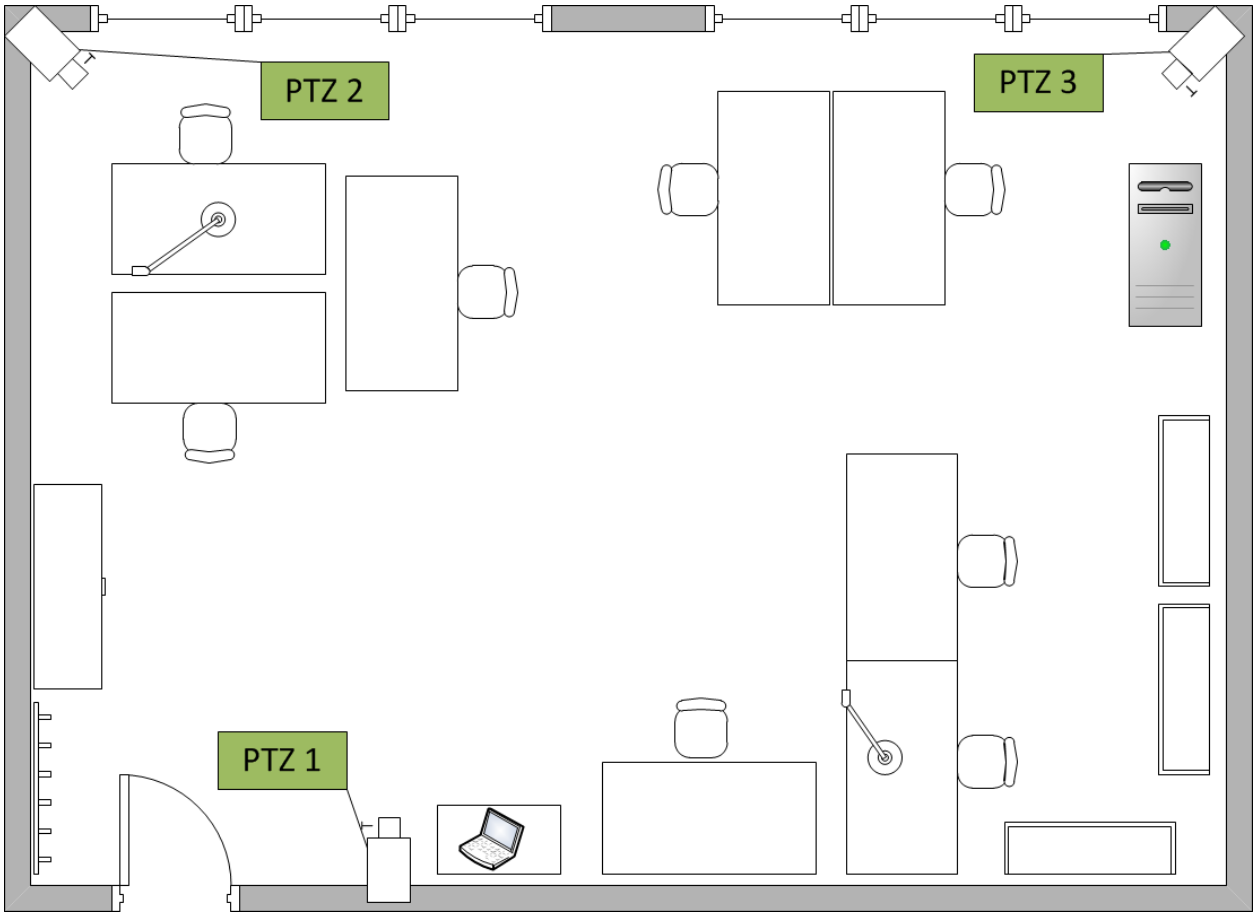


Ilustración 85 Plano del laboratorio GIAA

Como se puede observar en la Ilustración 86, existen 3 cámaras instaladas en el laboratorio. Para la prueba de concepto sólo será necesaria una de ellas. Dada la localización de las cámaras, la más idónea para usar el sistema es la macada como “PTZ 1”, ya que es la que mejor enfoca a la zona central de la habitación, donde se situarían los interlocutores a los que habría que localizar con Kinect.

Para facilitar la relación entre los ángulos calculados por el sensor Kinect y las direcciones a las que debe girar la cámara, se decide por comodidad situar el sensor Kinect justo debajo de la cámara PTZ 1. De este modo, cuando la aplicación detecte un interlocutor en un ángulo concreto, la correspondencia con el movimiento de la cámara PTZ necesario para apuntar a dicho interlocutor será más fácil de calcular.

Ilustración 86 Laboratorio GIIA

El equipo donde esté ejecutando la aplicación cliente debe situarse a corta distancia del sensor Kinect ya que se conecta a éste mediante conexión USB. Por tanto, el equipo portátil se instalará en una mesa cercana, como se muestra en la Ilustración 88.

Ilustración 87 Entorno de instalación de la prueba de concepto

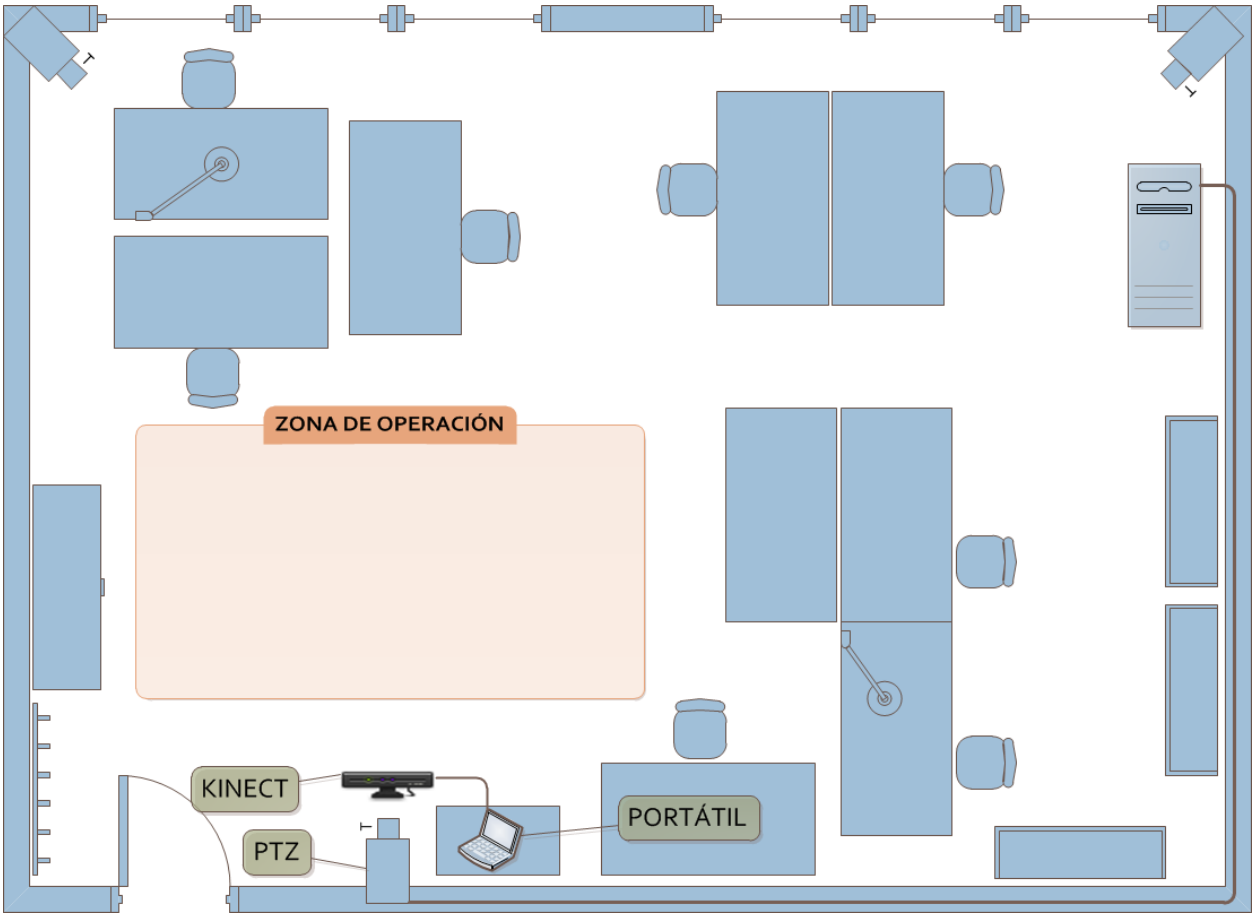


Ilustración 88 Diagrama de instalación de la prueba de concepto

En la ilustración 89, la zona marcada como “Zona de operación” marca el área donde los interlocutores deberán colocarse.

5.5. Programa servidor

Para la realización de este PFC se ha reutilizado este excelente programa diseñado por Álvaro Luis Bustamante y que interacciona con las cámaras PTZ del laboratorio. Entre sus funciones destacan:

- Control remoto de cámara PTZ por medio de comandos UDP para controlar la dirección, el zoom u otros comandos.
- Inicia o detiene un servicio de retransmisión *streaming* de las imágenes capturadas por la cámara PTZ.
- Inicia o detiene un servicio de recepción de comandos UDP para poder controlar la cámara desde aplicaciones externas.

Se usará por tanto desde la aplicación Kinect como interfaz de acceso a la cámara PTZ para obtener imágenes y para enviar comandos de forma que la cámara se mueva a la posición deseada.

La siguiente ilustración muestra la pantalla principal de esta aplicación.

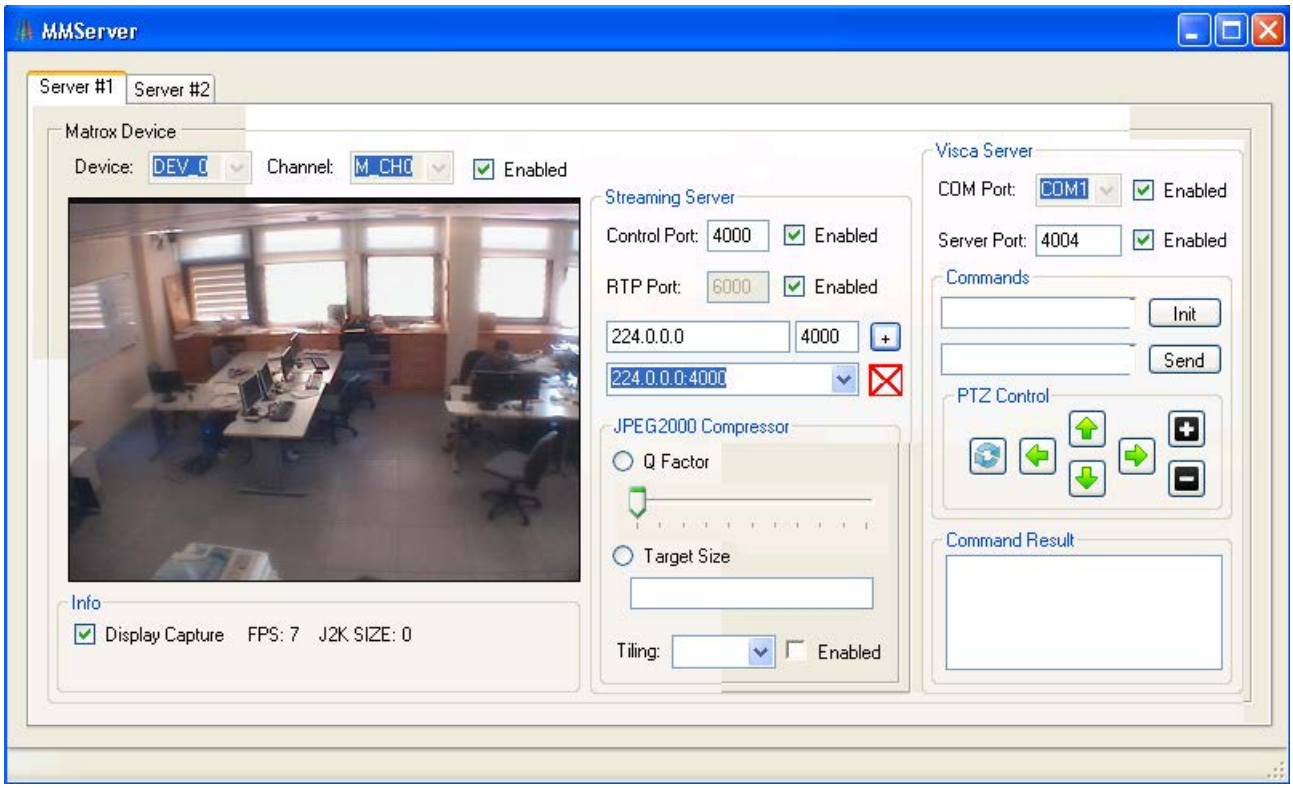


Ilustración 89 Aplicación servidor

Al iniciarse el servicio *streaming*, se empieza a emitir por una dirección *multicast* las imágenes capturadas por la cámara que esté activada. Desde el punto de vista de la aplicación cliente, únicamente es necesario realizar una conexión a esta dirección *multicast* y crear una sesión para empezar a recibir imágenes. Este proceso se detalla en el siguiente apartado.

5.6. Aplicación cliente con Kinect

Este es el apartado principal de este capítulo, ya que en él se describe la aplicación que forma el núcleo de la prueba de concepto. Como se ha mencionado ya, este programa usará el sensor Kinect a través del SDK de Kinect versión 1.5 para obtener flujos de datos y mostrárselos al usuario.

Los flujos de audio y seguimiento de esqueleto los obtendrá del sensor Kinect y el flujo de imagen lo obtendrá de la cámara PTZ. En este apartado se describirá cómo se han implementado las principales funciones del programa en el código fuente.

A grandes rasgos, la aplicación se compone de cuatro funciones principales:

- Localización de fuente sonora a partir de flujo de audio de Kinect.
- Gestión de la imagen obtenida a partir de flujo de imagen de la cámara RGB de Kinect.
- Gestión de la imagen obtenida de la cámara PTZ
- Panel de control de la aplicación

Estas cuatro funciones son las que definirán la estructura de la interfaz de usuario. Por tanto ésta estará dividida en cuatro apartados, uno destinado a cada función.

5.6.1. Interfaz de usuario principal

El proceso de desarrollo comienza creando un proyecto de Microsoft Visual Studio 2010, la herramienta de desarrollo básica para desarrollar aplicaciones para Windows. La tecnología a usar es **WPF (*Windows Presentation Foundation*),** que forma parte del framework .NET.

Una vez creado el proyecto se añaden las distintas librerías que el sistema va a necesitar (propias de .NET, librerías de terceros y del SDK de Kinect). WPF permite dividir el diseño de la interfaz de usuario mediante **el lenguaje declarativo XAML** de la lógica de la aplicación en C#. De esta forma se puede diseñar al mismo tiempo de forma paralela tanto la interfaz de usuario como la lógica del programa.

Además de XAML para el diseño de la vista y C# para la lógica general, es necesario mencionar otras biblioteca que se han usado en el desarrollo de esta aplicación. Para la grabación de vídeo se ha usado **Emgu CV**, una biblioteca desarrollada como un envoltorio de Open CV para el lenguaje C#, Visual Basic y Visual Basic C++, entre otras. Usando esta biblioteca seremos capaces de usar las funcionalidades de Open CV para grabación de vídeo.

Para establecer la conexión entre el programa servidor y cliente para la transmisión de imágenes capturadas por la cámara PTZ se usarán las funciones proporcionadas por la biblioteca **J2KClient.dll.** Esta librería implementa el algoritmo de compresión de imagen J2000 para retransmitir imágenes entre el programa servidor y el cliente.

Por último, para realizar el proceso de unión del vídeo generado (siendo la fuente Kinect o la cámara PTZ) y el audio grabado, se utilizará un programa externo denominado **FFmpeg** ampliamente utilizado por la comunidad para edición de vídeo y audio. Tiene multitud de opciones y su uso es complejo, pero aquí sólo se va a usar para unir las grabaciones de audio y video.

La interfaz de usuario principal que se ha diseñado se muestra a continuación:

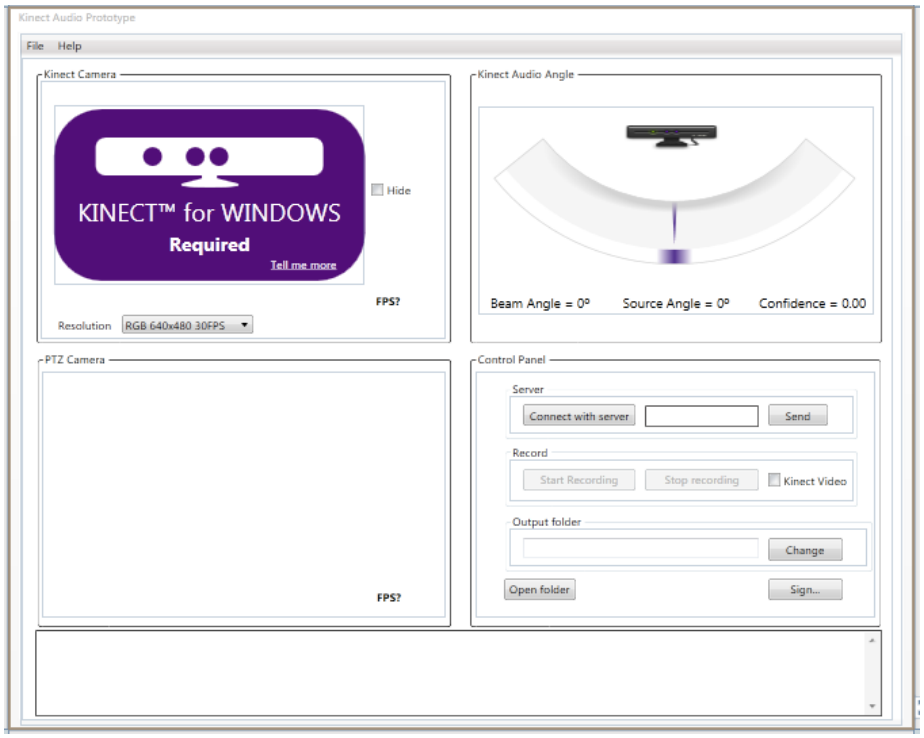


Ilustración 90 Diseño de la interfaz de usuario principal

El diseño de la interfaz de usuario de la aplicación en XAML es el siguiente:

```
<Window x:Class="PrototypeKinectApp.KinectAudioPrototype"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Kinect Audio Prototype" mc:Ignorable="d"    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
         xmlns:my="clr-namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinect.WpfViewers"
         Closing="Window_Closing" xmlns:my1="clr-namespace:PrototypeKinectApp"
         Icon="/PrototypeKinectApp;component/kinect.ico" ResizeMode="NoResize" d:DesignHeight="764"
         d:DesignWidth="955" SizeToContent="WidthAndHeight">
  <Grid Width="930" Height="732">
    <Grid Margin="1,26,0,6" Name="grid1" HorizontalAlignment="Left" Width="919">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="452*" />
        <ColumnDefinition Width="452*" />
      </Grid.ColumnDefinitions>
      <Grid.RowDefinitions>
        <RowDefinition Height="309*" />
        <RowDefinition Height="391*" />
      </Grid.RowDefinitions>
      <GroupBox Header="Control Panel" Name="groupBoxControl" Margin="13,0,0,98" Grid.Column="1"
                      BorderBrush="#FF171616" HorizontalAlignment="Left" Width="435" Grid.Row="1">
        <Grid Width="422" Height="260">
          <Button Content="Sign..." Height="23" HorizontalAlignment="Left" Margin="308,219,0,0" Name="buttonSign"
                     VerticalAlignment="Top" Width="79" Click="buttonSign_Click" />
          <GroupBox Header="Server" Height="64" HorizontalAlignment="Left" Margin="30,9,0,0" Name=
         "groupBox1" VerticalAlignment="Top" Width="374">
            <Grid>
              <Grid.ColumnDefinitions>
                <ColumnDefinition Width="291*" />
                <ColumnDefinition Width="41*" />
              </Grid.ColumnDefinitions>
              <Button Content="Connect with server" Height="23" HorizontalAlignment="Left" M
        argin=    "14,9,0,0" Name="buttonConnectServer" VerticalAlignment="Top" W
        idth="119" Click="buttonConnectServer_Click" />
              <TextBox Height="23" HorizontalAlignment="Left" Margin="143,10,0,0" Name=
                  "textBoxCommand" VerticalAlignment="Top" Width="120" BorderBrush="#FF171414" />
              <Button Content="Send" Height="23" HorizontalAlignment="Left" Margin="272,9,0,0" Name="
                  buttonSendCommand" VerticalAlignment="Top" Width="64" Click="
                  buttonSendCommand_Click" Grid.ColumnSpan="2" />
            </Grid>
          </GroupBox>
          <GroupBox Header="Record" Height="68" HorizontalAlignment="Left" Margin="30,76,0,0" Name="
         groupBox2" VerticalAlignment="Top" Width="374">
            <Grid>
              <Button Content="Start Recording" Height="23" Name="buttonRecord" Width="119" Click="
         buttonRecord_Click" Margin="14,10,0,0" HorizontalAlignment="Left" IsEnabled=
         "False" VerticalAlignment="Top" />
              <Button Content="Stop recording" Height="23" Name="buttonStopRecording" Margin=
                  "0,10,99,0" HorizontalAlignment="Right" Width="120" IsEnabled="False" Click="
                  buttonStopRecording_Click" VerticalAlignment="Top" />
              <CheckBox Content="Kinect Video" Height="16" HorizontalAlignment="Left" Margin="
                  272,14,0,0" Name="checkBoxKinectVideo" VerticalAlignment="Top" Checked="
                  checkBoxKinectVideo_Checked" Unchecked="checkBoxKinectVideo_Unchecked" />
            </Grid>
          </GroupBox>
                    <GroupBox Header="Output folder" Height="63" HorizontalAlignment="Left" Margin="
                  30,150,0,0" Name="groupBoxOutputFolder" VerticalAlignment="Top" Width="389">
            <Grid>
              <Button Content="Change" Height="24" HorizontalAlignment="Right" Margin="
                  0,9,26,0" Name="button1" VerticalAlignment="Top" Width="79" Click="
                  buttonSetOutputFolder" />
              <TextBox Height="22" HorizontalAlignment="Left" Margin="14,9,0,0" Name="
                  textBoxOutFolder" VerticalAlignment="Top" Width="249" IsReadOnly="True" F
         ontStyle="Italic" />
            </Grid>
          </GroupBox>
          <Button Content="Open folder" Height="23" HorizontalAlignment="Left" Margin="30,219,0,0"
         Name="buttonOpenOutputFolder" VerticalAlignment="Top" Width="75" Click="
         buttonOpenOutputFolder_Click" />
        </Grid>
      </GroupBox>
      <GroupBox Header="PTZ Camera" HorizontalAlignment="Left" Margin="16,0,0,98" Name="groupPTZVideo"
          Width="438" BorderBrush="#FF070606" Grid.Row="1">
        <Grid Height="260" Width="426">
          <Image Height="230" HorizontalAlignment="Left" Margin="13,14,0,0" Name="framePTZ"
                  Stretch="Fill" VerticalAlignment="Top" Width="307" />
```

```
        <Label Content="FPS?" FontWeight="Bold" Height="28" HorizontalAlignment="Left"
       Margin="348,226,0,0" Name="labelFPS2" VerticalAlignment="Top" Width="72" />
      </Grid>
    </GroupBox>
    <TextBox HorizontalAlignment="Left" Margin="15,296,0,4" Name="textBoxLog" Width="893" B
       orderBrush="#FF070707" VerticalScrollBarVisibility="Visible" TextChanged="textBoxLog_
       TextChanged" Grid.ColumnSpan="2" Grid.Row="1" />
    <GroupBox Header="Kinect Audio Angle" Margin="13,8,11,6" Name="groupKinectAngle" BorderBrush=
       "#FF211E1E" Grid.Column="1">
      <my1:AudioKinectControl x:Name="audioKinectControl1" Height="220" Width="417" />
    </GroupBox>
    <GroupBox Header="Kinect Camera" Margin="15,0,0,6" Name="groupKinectCamera" BorderBrush="#FF101010" H
       orizontalAlignment="Left" Width="439" Height="295" VerticalAlignment="Bottom">
      <Grid>
        <CheckBox Content="Hide" Height="16" HorizontalAlignment="Left" Margin="348,107,0,0" Name=
       "checkHideKinect" VerticalAlignment="Top" Checked="checkHideKinect_Checked" Unchecked="
       checkHideKinect_Unchecked" Width="54" />
        <ComboBox Height="19" HorizontalAlignment="Left" Margin="85,247,0,0" Name="comboBox1" V
       erticalAlignment="Top" Width="138" SelectedIndex="0" S
       electionChanged="comboBox1_SelectionChanged">
          <ComboBoxItem Content="RGB 640x480 30FPS" />
          <ComboBoxItem Content="RGB 1280x960 12FPS" />
          <ComboBoxItem Content="YUV 640x480 15FPS" />
        </ComboBox>
        <Label Content="Resolution" Height="28" HorizontalAlignment="Left" Margin="13,245,0,0" Name=
              "label1" VerticalAlignment="Top" />
        <Image Height="240" HorizontalAlignment="Left" Margin="13,3,0,0" Name="imageKinect" Stretch="
       Fill" VerticalAlignment="Top" Width="320" />
        <Label Content="FPS?" Height="28" HorizontalAlignment="Left" Margin="348,218,0,0" Name="
       labelFPS" VerticalAlignment="Top" Width="72" FontWeight="Bold" />
        <my:KinectSensorChooser Name="kinectSensorChooser1" Margin="13,25,85,57" />
      </Grid>
    </GroupBox>
  </Grid>
  <Menu Name="menu1" Margin="0,0,1,707" Height="23" VerticalAlignment="Bottom">
    <MenuItem Header="_File">
      <MenuItem Header="_New..." />
      <Separator />
      <MenuItem Header="_Open..." />
      <Separator />
      <MenuItem Header="_Save" />
      <MenuItem Header="_Save As..." />
      <Separator />
      <MenuItem Header="_Settings" />
      <Separator />
      <MenuItem Header="_Exit" Click="exit_Click" />
    </MenuItem>
    <MenuItem Header="_Help">
      <MenuItem Header="_User Manual" />
      <MenuItem Header="_Kinect Documentation" />
      <Separator />
      <MenuItem Header="_About" Click="MenuItem_Click" />
    </MenuItem>
  </Menu>
 </Grid>
</Window>
```

Lo importante de esta declaración, que para un iniciado puede parecer compleja, es que a interfaz está compuesta por secciones, cada una de ella formada por elementos: campos de texto, imágenes, y demás controles de usuario.

A continuación se describirá cada sección importante por separado.

5.6.2. Localización de fuente sonora

Para implementar esta funcionalidad se han usado los conocimientos adquiridos en el apartado 3.7 Guía de programación y el ejemplo de referencia Audio Basics.

Control de usuario AudioKinectControl

Este control se basa en el ejemplo incluido en *Kinect for Windows SDK 1.5* en el conjunto de ejemplos de referencia denominado *Kinect for Windows Developer Toolkit.* En la ilustración 92 se muestra esta aplicación.

Se ha partido de este ejemplo y de su código fuente proporcionado para crear un control de usuario modificando el ejemplo y editándolo a las necesidades de la prueba de concepto. Este control interactuará con Kinect para capturar el flujo de audio y, a través del SDK de Kinect, obtener los cambios en la localización de fuente sonora.

Ilustración 91 Aplicación de referencia Audio Basics

Se ha eliminado la representación de la señal digital del control y se ha modificado el tamaño del resto de elementos para que cupiera en la interfaz de usuario. La descripción XAML de este control es la siguiente:

```
<UserControl x:Class="PrototypeKinectApp.AudioKinectControl"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008" mc:Ignorable="d" Name="Root"
      Height="251" Width="400">
  <UserControl.Resources>
    <SolidColorBrush x:Key="MediumGreyBrush" Color="#ff6e6e6e"/>
    <Color x:Key="KinectPurpleColor">#ff52318f</Color>
  </UserControl.Resources>

  <Grid Name="layoutGrid" Height="215">
    <Grid.RowDefinitions>
      <RowDefinition Height="290*"/>
    </Grid.RowDefinitions>
    <Viewbox Stretch="Uniform" Margin="-88,-80,-85,0" Height="265" VerticalAlignment="Top">
      <Canvas Width="1" Height="0.52">
        <Path Data="M 0.1503,0.2832 L 0.228,0.2203 A 0.35,0.35 102 0 0 0.772,0.2203 L 0.8497,0.2832">
          <Path.Fill>
            <RadialGradientBrush Center="0.5 -0.32" RadiusX="0.65" RadiusY="1.1">
              <RadialGradientBrush.GradientStops>
                <GradientStop Color="LightGray" Offset="0"/>
                <GradientStop Color="LightGray" Offset="0.4"/>
```

```
              <GradientStop Color="WhiteSmoke" Offset="0.6"/>
              <GradientStop Color="WhiteSmoke" Offset="1"/>
            </RadialGradientBrush.GradientStops>
          </RadialGradientBrush>
        </Path.Fill>
      </Path>
      <Path Data="M 0.1270,0.3021 L 0.1503,0.2832 A 0.45,0.45 102 0 0 0.8497,0.2832 L 0.8730,0.302">
        <Path.Fill>
          <LinearGradientBrush StartPoint="0,0.5" EndPoint="1,0.5">
            <LinearGradientBrush.GradientStops>
              <GradientStop x:Name="sourceGsStart" Color="White"  Offset="0" />
              <GradientStop x:Name="sourceGsPre"  Color="White"  Offset="0.45" />
              <GradientStop x:Name="sourceGsMain"  Color="{StaticResource KinectPurpleColor}
                          " Offset="0.5" />
              <GradientStop x:Name="sourceGsPost"  Color="White"  Offset="0.55" />
              <GradientStop x:Name="sourceGsEnd"  Color="White"  Offset="1" />
            </LinearGradientBrush.GradientStops>
            <LinearGradientBrush.Transform>
              <RotateTransform x:Name="sourceRotation" CenterX="0.5" CenterY="0.0" Angle=
"0"></RotateTransform>
            </LinearGradientBrush.Transform>
          </LinearGradientBrush>
        </Path.Fill>
      </Path>
      <Path Data="M 0.495,0.35 L 0.505,0.35 L 0.5,0.44 Z">
        <Path.RenderTransform>
          <RotateTransform x:Name="beamRotation" CenterX="0.5" CenterY="0.0" Angle="0"/>
        </Path.RenderTransform>
        <Path.Fill>
          <LinearGradientBrush>
            <GradientStop Color="LightGray" Offset="0"/>
            <GradientStop Color="{StaticResource KinectPurpleColor}" Offset="0.5"/>
            <GradientStop Color="{StaticResource KinectPurpleColor}" Offset="1"/>
          </LinearGradientBrush>
        </Path.Fill>
      </Path>
      <Path Data="M 0.1270,0.3021 L 0.228,0.2203 A 0.35,0.35 102 0 0 0.772,0.2203 L 0.8730,0.3021"
         StrokeThickness="0.002" Stroke="LightGray" Canvas.Left="0" Canvas.Top="0" />
    </Canvas>
  </Viewbox>
  <Grid Margin="0,197,0,0" Height="16" VerticalAlignment="Top">
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="0*"/>
      <ColumnDefinition Width="133*"/>
      <ColumnDefinition Width="135*"/>
      <ColumnDefinition Width="132*"/>
      <ColumnDefinition Width="0*"/>
    </Grid.ColumnDefinitions>
    <TextBox Name="beamAngleText" FontSize="14" FontFamily="Segoe" HorizontalAlignment="Center"
      IsReadOnly="True" BorderThickness="0" Grid.Column="1" Width="126" Margin="7,0,0,-11">
        Beam Angle = 0</TextBox>
    <TextBox Grid.Column="2" Name="sourceAngleText" FontSize="14" FontFamily="Segoe" H
     orizontalAlignment="Center" IsReadOnly="True" BorderThickness="0" Width="139"
     Margin="5,0,123,-11" Grid.ColumnSpan="2">
          Source Angle = 0º</TextBox>
    <TextBox Grid.Column="3" Name="sourceConfidenceText" FontSize="14" FontFamily="Segoe"

HorizontalAlignment="Center" IsReadOnly="True" BorderThickness="0" Width="176" Margin=
                                                                        "11,0,-55,- 11"
Grid.ColumnSpan="2">Confidence = 0.00</TextBox>
    <Image Canvas.Left="0" Canvas.Top="0" Name="KinectIcon" Stretch="Fill" Source=
"/PrototypeKinectApp;component/Kinect.png" Margin="16,-212,26,169" Grid.Column="2" />
  </Grid>
 </Grid>
</UserControl>
```

En *AudioKinectControl.xaml* se definen todos los componentes que forman el control: campos de texto, áreas sombreadas, imágenes, etc. El resultado final es un control de usuario de WPF denominado *AudioKinectControl*. El aspecto final del control diseñado es el que se muestra en la siguiente ilustración:

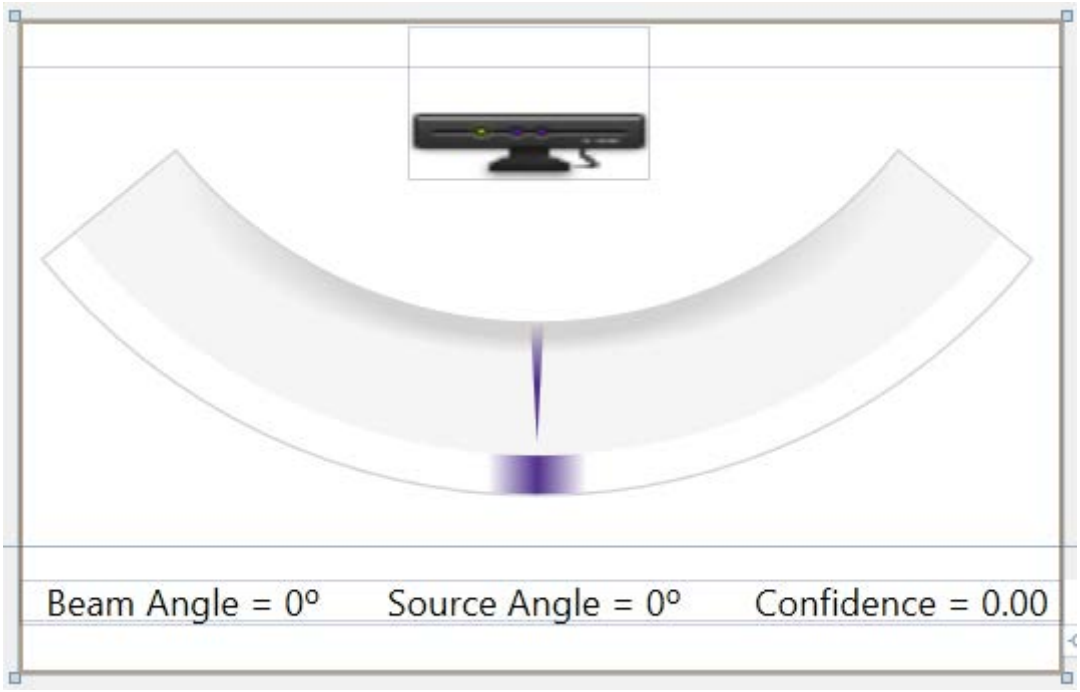


Ilustración 92 Diseño final del control de usuario AudioKinectControl

Este control es mostrado en la interfaz de usuario principal añadiendo una simple línea a su fichero XAML:

```
        <my1:AudioKinectControl x:Name="audioKinectControl1" Height="220" Width="417" />
```

La lógica del control de describe en el fichero *AudioKinectControl.xaml.cs.* Ya que no se trata de la aplicación principal no se va a mostrar en este documento todo el contenido de este fichero. Sí se muestra, sin embargo, la función principal encargada de notificar a la aplicación principal la modificación del valor del ángulo calculado

Estas son las dos funciones principales del control AudioKinectControl:

```
/// <summary>
/// Handles event triggered when audio beam angle changes.
/// </summary>
/// <param name="sender">object sending the event.</param>
/// <param name="e">event arguments.</param>
private void AudioSourceBeamChanged(object sender, BeamAngleChangedEventArgs e)
{
  beamRotation.Angle = -e.Angle;

  string beamAngle = e.Angle.ToString("0", CultureInfo.CurrentCulture);
  beamAngleText.Text = string.Format(CultureInfo.CurrentCulture, Properties.Resources.BeamAngle,
    beamAngle);

  // PRINCIPAL MODIFICACIÓN: CUANDO EL ÁNGULO CAMBIA, SE NOTIFICA EL CAMBIO DE ESTA PROPIEDAD PARA QUE
  // LA APLICACIÓN PRINCIPAL PUEDA ENVIAR COMANDOS A LA CÁMARA PTZ
  OnPropertyChanged(beamAngle);
}
```

La función AudioSourceBeamChanged lanza una evento de notificación cuando el sensor Kinect detecta que ha habido un cambio en la localización de la fuente de audio. En concreto, muestra la zona (*beam*) de la que el sonido proviene. Recordemos que Kinect reconoce hasta 11 secciones de 10 grados cada una desde -50 grados a +50 grados. Además, muestra en el control este cambio en un campo de texto.

Cuando la interfaz principal recibe esta notificación, envía una orden de movimiento a la cámara PTZ a través del programa servidor para que la cámara pase a enfocar la posición calculada por Kinect de la que proviene el sonido. Para enviar este comando se utiliza la función sendCommand:

```
private void sendCommand(string command)
{
  string[] words = command.Split(' ');

  if (command.Equals(""))
  {
    this.textBoxCommand.FontStyle = FontStyles.Italic;
    this.textBoxCommand.Text = "Invalid command";
    return;
  }

  try
  {
    udpClient.Connect(serverIP, serverPort);
    // Sends a message to the host to which you have connected.
    Byte[] sendBytes = Encoding.ASCII.GetBytes(command);
    udpClient.Send(sendBytes, sendBytes.Length);
    this.logWriteLine("Command sent: "+command);
  }
  catch (Exception ex)
  {
    Console.WriteLine(ex.ToString());
  }
}
```

De esta forma, la cámara siempre estará apuntando a la dirección que vaya calculado Kinect y únicamente tendremos que centrarnos en almacenar las imágenes que nos envíe la cámara.

El valor del *beam* calculado se puede obtener del campo Angle de los parámetros del evento, BeamAngleChangedEventArgs.

```
/// <summary>
/// Handles event triggered when sound source angle changes.
/// </summary>
/// <param name="sender">object sending the event.</param>
/// <param name="e">event arguments.</param>
private void AudioSourceSoundSourceAngleChanged(object sender, SoundSourceAngleChangedEventArgs
e)
{
  // Maximum possible confidence corresponds to this gradient width
  const double MinGradientWidth = 0.04;

  // Set width of mark based on confidence.
  // A confidence of 0 would give us a gradient that fills whole area diffusely.
  // A confidence of 1 would give us the narrowest allowed gradient width.
  double halfWidth = Math.Max((1 - e.ConfidenceLevel), MinGradientWidth) / 2;

  // Update the gradient representing sound source position to reflect confidence
  this.sourceGsPre.Offset = Math.Max(this.sourceGsMain.Offset - halfWidth, 0);
```

```
  this.sourceGsPost.Offset = Math.Min(this.sourceGsMain.Offset + halfWidth, 1);

  // Rotate gradient to match angle
  sourceRotation.Angle = -e.Angle;

  sourceAngleText.Text = string.Format(CultureInfo.CurrentCulture,
Properties.Resources.SourceAngle, e.Angle.ToString("0", CultureInfo.CurrentCulture));
  sourceConfidenceText.Text = string.Format(CultureInfo.CurrentCulture,
Properties.Resources.SourceConfidence, e.ConfidenceLevel.ToString("0.00",
CultureInfo.CurrentCulture));
}
```

La función AudioSourceSoundSourceAngleChanged no notifica el cambio del ángulo obtenido a la interfaz principal pero sí la muestra en el control y produce una rotación en el indicador del control. Además, dependiendo del grado de confianza devuelto se dibuja en la misma dirección del indicador un gradiente. Este gradiente será más ancho cuanto menor sea el valor de la confianza. Se trata de una forma muy visual y sencilla de representar la precisión del valor del ángulo.

5.6.3. Grabación de audio

Para generar el vídeo final se usará como fuente de audio el conjunto de micrófonos de Kinect. Esta señal acústica se almacenará en un fichero en formato WAV y cuando finalice la grabación será unido al vídeo generado por Emgu CV.

Para grabar audio del flujo de datos de sonido de Kinect se modificará el programa de ejemplo *Record Audio* incluido en el SDK v1.0. La función que realiza la grabación se denomina RecordAudio y se describe a continuación:

```
/// <summary>
/// Saves audio from kinect audio stream to WAV file
/// </summary>
private void RecordAudio()
{
  this.logWriteLine("Audio recording started...");
  if (kinectSensor == null)
  {
    return;
  }
  this.audioFileName = outputFolder + audioFileNameOiginal + "_" +
DateTime.Now.ToString("yyyyMMddHHmmss") + ".wav";
  StringWriter sw = new StringWriter(CultureInfo.InvariantCulture);
  // int recordingLength = (int) _amountOfTimeToRecord * 2 * 16000;
  byte[] buffer = new byte[4096];
  int recordingLength = 0;

  Thread.CurrentThread.Priority = ThreadPriority.Highest;
  using (var fileStream = new FileStream(audioFileName, FileMode.Create))
  {
    WriteWavHeader(fileStream);

    // Simply copy the data from the stream down to the file
    int count;
    while (readStream && ((count = audioStream.Read(buffer, 0, buffer.Length)) > 0))
    {
      //this.checkBeamAngle();

      for (int i = 0; i < buffer.Length; i += 2)
      {
        short sample = (short)(buffer[i] | (buffer[i + 1] << 8));
```

```
      }

      fileStream.Write(buffer, 0, count);
      recordingLength += count;
    }
    UpdateDataLength(fileStream, recordingLength);
    this.logWriteLine("Audio recording finished! File saved: " + audioFileName);

    this.finAudio = true;
  }
}
```

La función almacena los datos del flujo en un fichero al que se le ha dado el formato específico WAV mediante información de cabecera. No es necesario usar bibliotecas externas ya que los datos se almacenan tal cual se reciben del sensor, en bruto.

Cuando la grabación finalice, en la carpeta de salida de la aplicación se habrá generado un fichero en formato WAV identificado con la fecha de grabación.

5.6.4. Imagen de Kinect

Obtener imágenes a partir del flujo de datos de la cámara RGB de Kinect es un proceso sencillo. Simplemente se asocia una función al evento que lanza el SDK cuando el sensor tiene lista la siguiente imagen. Esta función transforma los datos del flujo en un control de usuario de tipo *Image* y muestra la imagen en la interfaz de usuario.

El proceso es casi idéntico al que se describió en la Guía de Programación. A continuación se muestra el evento que controla este proceso:

```
/// <summary>
/// This event fires when Color stream is ready
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
///
private void ColorImageReady(object sender, ColorImageFrameReadyEventArgs e)
{
  using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
  {
    if (colorFrame == null)
      return;
    try
    {
      colorFrame.CopyPixelDataTo(pixels);
      //Muestra en IU
      this.imageKinect.Source = BitmapSource.Create(colorFrame.Width, colorFrame.Height, 96, 96,
          PixelFormats.Bgr32, null, pixels, colorFrame.Width * 4);
    }
    catch (Exception)
    {
      //Drop frame
      this.droppedFrames++;
      return;
    }

        // Para grabar video usando imágen de Kinect
    if (recording && grabarConKinect)
    {
      if (bmap == null)
      {
        this.logWriteLine("Kinect video recording started...");
```

```
        bmap = new Bitmap(colorFrame.Width, colorFrame.Height,
          System.Drawing.Imaging.PixelFormat.Format32bppRgb);
      }

      bmapdata = bmap.LockBits(rect, ImageLockMode.WriteOnly,
        System.Drawing.Imaging.PixelFormat.Format32bppRgb);
      Marshal.Copy(pixels, 0, bmapdata.Scan0, colorFrame.PixelDataLength);
      bmap.UnlockBits(bmapdata);
      Image<Bgr, byte> ImageFrame = new Image<Bgr, Byte>(bmap);
      vw.WriteFrame<Bgr, Byte>(ImageFrame);
      ImageFrame.Dispose();
    }
    FPS_Count++;
  }
}
```

El evento transforma el flujo de datos en una estructura de datos capaz de ser tratada, un array de bytes, mediante la función `CopyPixelDataTo()`. Después es necesario transformar este array en un control de usuario visualizable en la interfaz de usuario. Por comodidad, este control es *Image*. La función asigna como fuente del control un objeto *BitmapSource* creado a partir de las características de imagen de Kinect: el formato de datos es Bpp32, lo que significa que cada pixel del flujo de datos está representado por 4 bytes de datos o 32 bits. Esta información, junto con el ancho, largo y tamaño de fila de imagen (en inglés, *stride*), es necesaria para crear el objeto *BitmapSource*.

La última sección sólo es válida si el usuario desea grabar vídeo usando como fuente Kinect en vez de la cámara PTZ. En el caso de que esté activada la grabación usando como fuente Kinect, se entraría en esta sección y se guardaría en fichero usando las funciones de EmguCV de la clase *VideoWriter*.

Estas y otras opciones se explicarán más adelante en el apartado *Panel de Control*.

5.6.5. Imagen de la cámara PTZ

El flujo de datos de imagen capturado por la cámara PTZ del laboratorio es enviada a la aplicación cliente por medio de una conexión en red. La aplicación servidor inicia un servicio de *streaming* al que el cliente se une y empieza a capturar estas tramas mediante una función que es invocada cada vez que el cliente recibe una nueva imagen.

La siguiente función es la encargada de gestionar la lógica que se ejecuta cuando el usuario presiona el botón de la interfaz de usuario para conectar con el servidor:

```
    /// <summary>
    /// Inicia la sesión de transferencia de frames con el servidor
    /// </summary>
    private void buttonConnectServer_Click(object sender, RoutedEventArgs e)
    {
      this.buttonConnectServer.Content = "Connecting...";
      this.buttonConnectServer.IsEnabled = false;
      this.logWriteLine("C# console...");

      if (J2KClient.StartSession(4000))
      {
        sesionActive = true;
        this.logWriteLine("RTP Session.... OK");

        mImageCallBack = ImageCallback;
        J2KClient.SetImageCallback(mImageCallBack);

        mNewSourceCallBack = NewSourceCallback;
        J2KClient.SetNewSourceCallback(mNewSourceCallBack);

        if (J2KClient.JoinMulticastGroup("224.0.0.0".ToCharArray()))
```

```
        {
          this.logWriteLine("Multicast Join.... OK");
        }
      }
    }
```

El programa establece una nueva conexión con el servidor en el puerto 4000 a través de la dirección 224.0.0.0 y establece que cada nueva llegada de imagen del servidor debe ser tratada por la función `ImageCallback`, que se describe a continuación:

```
/// <summary>
/// Procesa en hilos independintes recepciones de nuevas frames del servidor
/// </summary>
///
List<Image<Bgr, Byte>> _videoArray = new List<Image<Bgr, Byte>>();
private void ImageCallback(UInt32 sourceID, IntPtr image, UInt32 imageSize, UInt16 width, UInt16 height,
UInt16 components, UInt32 timestamp)
{
  try
  {
    if (recording && !grabarConKinect)
    {
      Bitmap bitit = new Bitmap(width, height, width * 3,
        System.Drawing.Imaging.PixelFormat.Format24bppRgb, image);

      Image<Bgr, Byte> ImageFrame2 = new Image<Bgr, Byte>(bitit);
      _videoArray.Add(ImageFrame2);
    }

    BitmapSource bit = BitmapSource.Create(width, height, 96, 96, PixelFormats.Bgr24, null, image,
      width * height * 3, width * 3);

    if (bit.CanFreeze)
    {
      bit.Freeze();
    }

    // Invocation required
    framePTZ.Dispatcher.Invoke(DispatcherPriority.Normal,
      (UpdateTheUI)delegate(BitmapSource origen) { framePTZ.Source = origen; }, bit);
  }
  catch (Exception w)
  {
    this.logWriteLine("Cuidado: " + w.Message);
  }
  finally
  {
    this.FPS_Count2++;
  }
}
```

Esta función es invocada cada vez que se recibe una nueva imagen del servidor. Para cada vez, por defecto (si no está activada la grabación usando Kinect) se crea un objeto de tipo `Image<Bgr, Byte>` para almacenar el frame y se añade a la lista de frames que serán escritas en fichero mediante EmguCV.

Además, siempre se muestra en la sección denominada *PTZ Camera* esta imagen al usuario usando un control de tipo *Image.* Este proceso, simple en teoría, ha sido uno de los que más tiempo ha requerido ya que es necesario acceder a la interfaz principal mediante el hilo principal, y cada llamada a `ImageCallback` se realiza por hilos independientes. En el apartado *Consideraciones finales* se habla de esta complicación.

Ilustración 93 Imagen de cámara PTZ

Como se puede observar, la velocidad con la que el servidor retransmite las imágenes de la cámara PTZ es más o menos de 17 imágenes por segundo. El tiempo de procesamiento es casi despreciable: como mucho se pierde una imagen por segundo.

5.6.6. Grabación de vídeo

En los apartados anteriores hemos descrito los componentes individuales necesarios para crear nuestro fichero de vídeo: imagen proveniente desde Kinect o desde cámara PTZ (siempre enfocando a la fuente de audio) y audio proveniente del sensor Kinect. En este apartado se define el proceso de generación del fichero de vídeo a partir de estas fuentes.

Cuando el usuario inicia el proceso de grabación, se ejecuta la función `buttonRecord_Click`:

```
/// <summary>
/// Inicia la grabación principal
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void buttonRecord_Click(object sender, RoutedEventArgs e)
{
  videoFileName = outputFolder + videoFileNameOriginal + "_" + DateTime.Now.ToString("yyyyMMddHHmmss") +
".avi";

  if (vw == null)
  {
    try
    {
      if (grabarConKinect)
      {
        vw = new VideoWriter(videoFileName,
          (int)-1, //Permite seleccionar códec de vídeo
          25,
          kinectWidth,
          kinectHeight,
          true);
      }
      else      {
        vw = new VideoWriter(videoFileName,
```

```
            0,
            18,
            768,
            576,
            true);
        }
      }
      catch (Exception)
      {
        this.logWriteLine("Please select the video codec");
        return;
      }
      finally
      {
        if (vw != null)
        {
          this.msInicio = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
          this.logWriteLine("Inicio(ms)= " + msInicio);

          this.buttonRecord.Content = "Recording";
          this.buttonRecord.IsEnabled = false;
          this.buttonStopRecording.IsEnabled = true;
          this.recording = true;

          //Start recording audio on new thread
          var t2 = new Thread(this.RecordAudio);
          t2.Start();
        }
      }
    }
}
```

Esta función inicia el proceso de grabación. Primero inicia el objeto de tipo *VideoWriter* según si la fuente proviene de Kinect o la cámara PTZ ya que cada uno tiene una velocidad de imágenes por segundo diferente. Después de imprimir información de control en la ventana de log de la interfaz de usuario, cambia el valor de una variable de control de forma que puedan empezar a almacenarse las imágenes e inicia un nuevo hilo para empezar a grabar audio.

Empiezan por tanto los procesos de grabación de audio y vídeo por separado. Cuando el usuario desee finalizar la grabación presiona el botón destinado a tal fin en el panel de control y se ejecuta la función `buttonStopRecording_Click`:

```
/// <summary>
/// Finaliza la grabación de video y audio
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void buttonStopRecording_Click(object sender, RoutedEventArgs e)
{
  this.msFin = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
  this.logWriteLine("Fin(ms)= " + msFin);
  this.logWriteLine("Recording time(ms)= " + (msFin - msInicio));
  this.buttonStopRecording.IsEnabled = false;
  this.buttonRecord.IsEnabled = true;
  this.buttonRecord.Content = "Start recording";
  this.recording = false;

  lock (bloqueador)
  {
    readStream = false;
  }
  for (int i = 0; i < _videoArray.Count(); i++)
    vw.WriteFrame<Bgr, Byte>(_videoArray[i]);
  vw.Dispose();
  this.vw = null;

  this.logWriteLine("Video recording finished! File saved: " + videoFileName);
  this.finVideo = true;
```

```
  //Start muxing audio and video on new thread
  var t = new Thread(this.muxing);
  t.Start();
}
```

Se finaliza la grabación de audio y vídeo alterando las variables lógicas que controlan el proceso. Para el caso de la grabación de audio, se ha generado un fichero en la carpeta de salida que contiene la grabación; para el vídeo, es necesario generar el fichero ya que lo se han ido almacenando imágenes en memoria. Lo que se hace ahora por tanto es usar un objeto de tipo *VideoWriter* de Emgu CV y almacenar cada imagen en un fichero. Este fichero también se escribe en la carpeta de salida.

Por último, un nuevo hilo ejecuta la función `muxing` encargada a añadir al video generado la grabación de audio:

```
/// <summary>
/// This method merges together audio stream (WAV file) and video stream (AVI file) using FFMPEG
/// </summary>
private void muxing()
{
  while (!finAudio && !finVideo){
    this.logWriteLine("Waiting for audio to finish...");
    Thread.Sleep(500);
  }
  this.logWriteLine("Joining audio and video");

  outputFileName = outputFolder + outputFileNameOriginal + "_" +
DateTime.Now.ToString("yyyyMMddHHmmss") + ".avi";

  string parameters = "-i " + audioFileName + " -i " + videoFileName + " -acodec copy -vcodec
copy " + outputFileName;
  Process.Start("ffmpeg.exe", parameters);

  this.logWriteLine("Joined! File saved: " + outputFileName);
}
```

Después de confirmar ha terminado la grabación de audio (por si acaso se produce algún retardo se realiza espera activa), se invoca al programa FFMPEG con las grabaciones de vídeo y audio. El programa une la grabación de audio en formato WAV al fichero de vídeo en formato AVI y genera un nuevo vídeo que almacena en la carpeta de salida.

Este fichero tiene como nombre ***recording_FechaDeGrabacion.avi*** y es el **vídeo resultado de la aplicación.**

5.6.7. Panel de control

El panel de control de la aplicación se sitúa abajo a la derecha en la interfaz de usuario, y su aspecto es el siguiente:

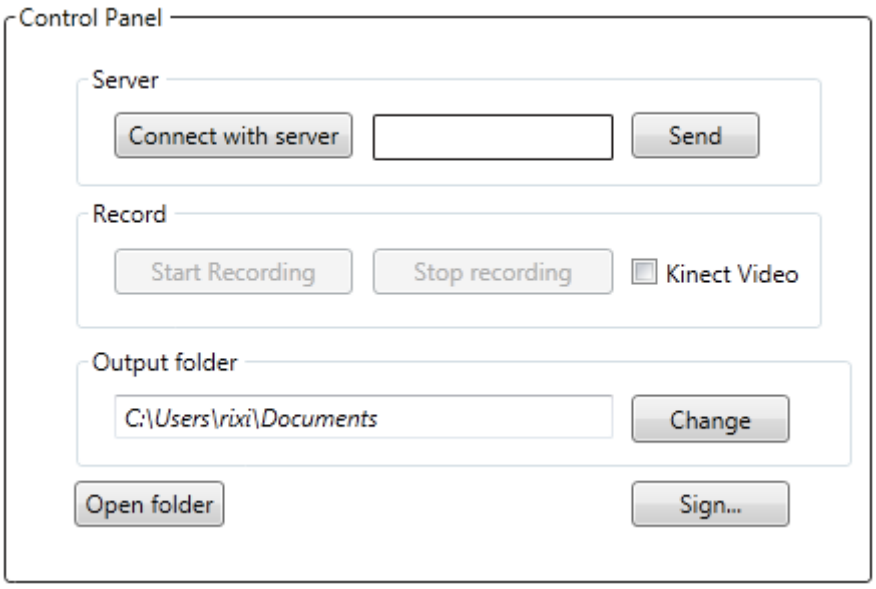


Ilustración 94 Panel de control de la aplicación

Contiene los medios para que el usuario interaccione con la aplicación de forma que pueda controlar el proceso de generación del vídeo resultado. Para ello, y tal como se describe en el apartado *Manual de usuario*, se definen los controles que permiten:

- Iniciar la conexión con el servidor.
- Enviar comandos al servidor para controlar la cámara: esta funcionalidad se controla internamente, de modo que el sistema automáticamente envía nuevas direcciones a la cámara.
- Activar la grabación de vídeo usando como fuente Kinect en vez de la cámara PTZ.
- Modificar la carpeta de salida del fichero de vídeo.
- Abrir la capeta de salida.
- Acceder al programa XolidoSign para realizar firmado digital.
- Iniciar o detener el proceso de grabación de vídeo.

Además de estas opciones, también es posible modificar el comportamiento de la aplicación usando controles situados fuera del panel de control. Para la representación de la imagen capturada por Kinect, es posible modificar:

- Resolución de la imagen. Las posibilidades son 640x480 a 30 FPS y 1280x960 a 12 FPS para formato RGB y 640x480 a 15 FPS en formato YUV.
- Opción de mostrar u ocultar esta captura para mejorar el rendimiento de la aplicación.

Estas opciones se muestran en la siguiente ilustración.

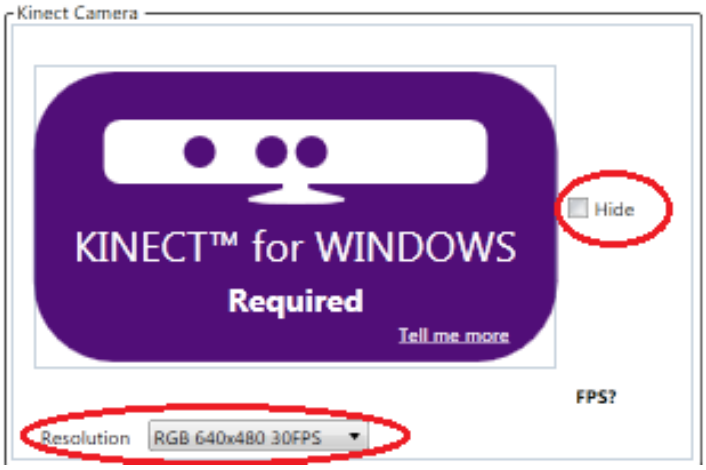


Ilustración 95 Opciones sobre captura de imagen desde Kinect

5.6.8. Firma digital

Una vez generada la video acta, queremos firmarla digitalmente para asegurar su autenticidad e integridad. De esta forma podrá asegurarse que no ha sido modificado el vídeo y que el autor del mismo es quien dice ser.

La firma digital es una herramienta muy utilizada actualmente para realizar operaciones que requieran seguridad y de esa forma garantizar autenticidad e integridad sobre ficheros electrónicos. Antes de explicar cómo se ha enfocado esta técnica en la prueba de concepto, analicemos primero sus bases.

Definición

Una firma digital es un esquema matemático que sirve para demostrar la autenticidad de un mensaje digital, que puede ser por ejemplo un documento electrónico. Una firma digital da al destinatario seguridad de que el mensaje fue creado por el remitente (autenticidad de origen), y que no fue alterado durante la transmisión (integridad). Las firmas digitales se utilizan comúnmente para la distribución de software, transacciones financieras y en otras áreas donde es importante detectar la falsificación y la manipulación.

La firma digital consiste en un método criptográfico que asocia la identidad de una persona o de un equipo informático, al mensaje o documento. En función del tipo de firma, puede además asegurar la integridad del documento o mensaje.

La firma electrónica, como la firma hológrafa (autógrafa, manuscrita), puede vincularse a un documento para identificar al autor, para señalar conformidad (o disconformidad) con el contenido, para indicar que se ha leído y, en su defecto mostrar el tipo de firma y garantizar que no se pueda modificar su contenido.

Terminología

- Una **firma digital** es una cadena de caracteres asociada a un mensaje digital que garantizan la autenticidad del origen e integridad del mismo. Para generarla se utilizan métodos criptográficos.
- La **firma electrónica** es un término de naturaleza fundamentalmente legal y más amplio desde un punto de vista técnico, ya que puede contemplar métodos no criptográficos, por ejemplo puede tratarse de una firma electrónica escrita.
- Un **algoritmo de generación de firma digital** es un método para producir firmas digitales.

- Un **algoritmo de verificación de firma digital** es un método que permite la verificación de que una firma digital es auténtica.
- Un **esquema o mecanismo de firma digital** consiste en un algoritmo de generación de firma y su algoritmo de verificación asociado.

Elección del método

Sobre esta base, y enfocándonos en firmar digitalmente nuestro fichero de vídeo, se ha realizado una proceso de investigación para dilucidar cómo incluir en la prueba de concepto esta funcionalidad y de esta forma poder asegurar la integridad y autenticidad del fichero de vídeo resultante.

Las posibilidades que se estudiaron para incluir la funcionalidad fueron dos:

- Incluirla en código, usando librerías .NET.
- Incluirla mediante un software de terceros ya implementado.

El primer método requería de una gran inversión en tiempo para aprender a manejar los procedimientos específicos al lenguaje de programación para poder realizar el proceso primero de firma con certificado y después de verificación.

El segundo método proporcionaba las ventajas de estar ya implementado y probados por expertos en el área. Ante estas dos opciones, y siendo esta función un añadido al objetivo principal de este PFC, hizo que se escogiera la segunda opción.

Elección de software

Una vez elegido el método, es necesario realizar un estudio de las posibilidades y escoger la que más nos convenga. Tras una concienzuda investigación de software *open source* para firmado digital de ficheros, se encontraron dos potenciales programas que cumplían los siguientes requisitos necesarios:

- Debe ser capaz de obtener la firma digital de un fichero independientemente de su formato. Esto es debido a que la mayoría del software existente para este cometido se centra en la firma de documentos PDF.
- Debe permitir el uso de certificados digitales.
- Debe ser capaz de realizar el proceso de verificado de firma digital dado un fichero y su firma.
- Debe ser *open source.* De este modo se distribuye con licencia libre y su uso no necesita de la adquisición de una licencia.

Los dos resultados encontrados fueron las aplicaciones **Sinadura Desktop** y **XolidoSign**.

XolidoSign es una aplicación de escritorio gratuita para firmar electrónicamente y realizar sellado de tiempo sobre cualquier tipo de fichero. Su última versión es la 2.1.0.10.

Sus funcionalidades incluyen:

- Uso de certificados digitales para firmado de ficheros.
- Verificación de firma para los formatos PKCS7 / CMS / CAdES / XMLDSig / XAdES de firma digital y Sellos de tiempo RFC 3161 y XML OASIS TST.
- Incluye soporte de librerías PKCS11 estándar de acceso a tarjetas criptográficas inteligentes.
- Incluye sellado de tiempo para dejar constancia de la existencia de un archivo en un momento determinado a través de servidores gratuitos.
- Incluye soporte a arquitecturas Windows de 64 bits (Windows 200, XP, Vista y 7).
- Idiomas: Inglés y Español.

- Incluye *drag and drop* de ficheros (arrastrarlos con el puntero a la interfaz de usuario).
- Preparado para usar con DNI electrónico.
- Puede firmar varios ficheros en una misma ejecución.
- Gran abanico de opciones.

Su interfaz de usuario principal se puede observar en la Ilustración SSS. Es sencilla de usar ya que expone las tres opciones principales de modo destacado (firmar, verificar y sello de tiempo).

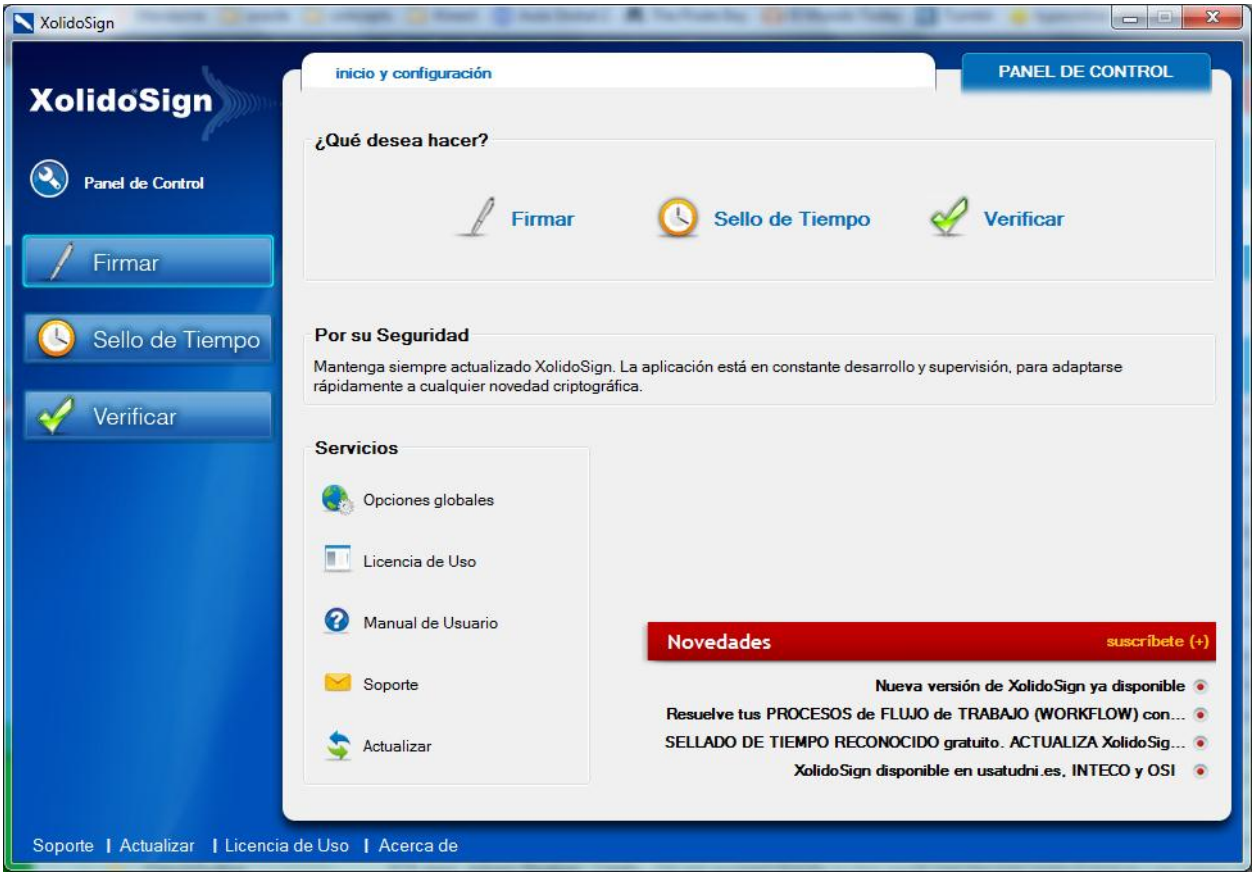


Ilustración 96 Aplicación XolidoSign

Sinadura es un proyecto open source orientado a ofrecer productos y servicios para la identidad digital y firma electrónica tanto para particulares como empresas y Administraciones Públicas. Ofrece herramientas de software, servicios y soporte a la comunidad.

La herramienta más conocida del proyecto de llama **Sinadura Desktop**. Es una aplicación de escritorio multiplataforma para la firma digital de cualquier tipo de archivo. El software afirma garantizar la integridad, identidad y el no repudio en cualquier archivo, como pueden ser nóminas, contratos, facturas o certificaciones en archivos de texto, canciones en archivos de sonido o videoclips en archivos de vídeo.

La última versión estable es la 3.0. Sus funcionalidades incluyen:

- Capacidad de firma de varios documentos al mismo tiempo.
- Soporta las plataformas Windows, GNU/Linux y MAC OS X.
- Requiere de máquina virtual de Java.
- Permite incorporar una marca de agua a documentos.

- Firma electrónica.
- Soporta Inglés, Castellano y Euskera.
- Verificación de firma.
- Compatible con los principales certificados.
- Fuerte apoyo por parte de la comunidad.

En la siguiente ilustración se muestra la interfaz de usuario de Sinadura:

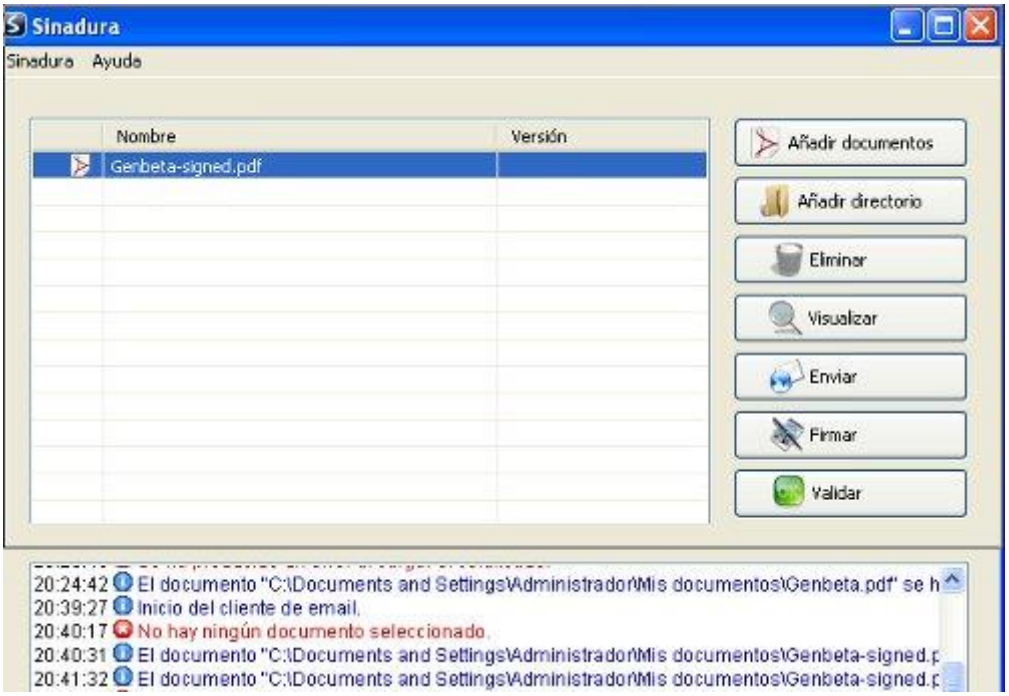


Ilustración 97 Interfaz de usuario de Sinadura

Su interfaz de usuario muestra las opciones principales y un log de eventos.

Conclusión

La conclusión a la que se ha llegado es que sin lugar a dudas **XolidoSign** es, de las dos, la herramienta más estable. Se descargaron las dos aplicaciones y se probó su funcionamiento.

Analicemos primero Sinadura. De entrada su instalación, tal y como avisa en la página oficial, puede provocar errores. En mi opinión esta afirmación se queda muy corta. La primera vez que se intentó instalar en un sistema Windows el programa instalador se quedaba bloqueado sin explicación, la siguiente vez con la versión de 64 bits arrojaba un error de Java (era necesario la versión de 64 bits de Java). Cuando se solucionó, el programa no se iniciaba, enviando al usuario un fallo por excepción. Como último recurso antes de descartarlo por no usable se probó a instalar en un sistema Linux. No sin errores, se consiguió lanzar la aplicación.

Para mayor consternación, era necesario configurar ciertos parámetros en un fichero de texto dentro de la carpeta de instalación. Continuamente el programa fallaba su ejecución sin previo aviso. Pese a que finalmente se consiguió satisfactoriamente realizar la firma de un fichero de vídeo (caso que nos ocupa) y posterior validación, se considera que **es una aplicación no usable y que requiere una gran depuración de errores** antes de ser distribuida al público.

A continuación se probó el software XolidoSign. Nada que ver con el anterior, este software se instala y ejecuta sin problemas (como se debe esperar de cualquier producto software que ha superado la fase beta). Sin ningún error se

consigue con unos cuantas interacciones de usuario seleccionar los ficheros a firmar, se firma y se valida. La validación es incluso más sencilla ya que tiene la capacidad de automáticamente detectar la firma si está en el mismo directorio que el fichero firmado. Se trata de un sistema mucho más estable y probado que Sinadura.

En vista de este análisis, la herramienta **XolidoSign** es la mejor alternativa de código libre para realizar firma y verificación de ficheros usando certificados digitales, razón por la que será utilizado en esta prueba de concepto.

Incorporación en la aplicación

Ya que toda la funcionalidad de firma y verificación de firma digital se va a implementar mediante una librería de terceros, simplemente se ha añadido un control de usuario de tipo botón en la interfaz de usuario que ejecute el programa XolidoSign.

Este botón está en la sección *Control Panel* de la interfaz principal y está etiquetado con el texto *"Sign…"*.

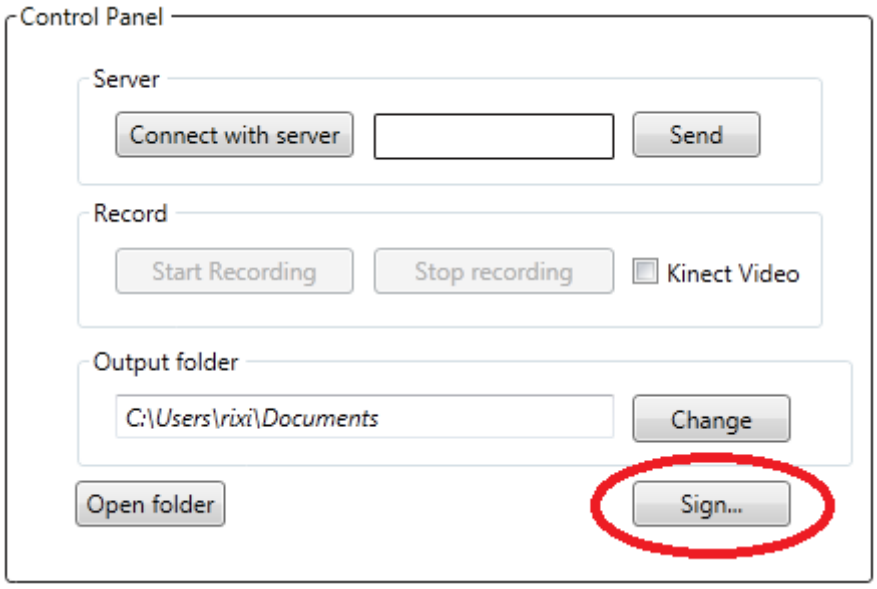


Ilustración 98 Ubicación de enlace a XolidoSign para firma digital desde interfaz de usuario

5.6.9. Consideraciones finales

Del proceso de desarrollo de esta aplicación, me gustaría comentar algunos aspectos que han llevado más trabajo.

Sin duda la mayor parte del tiempo consumido en actividades que podríamos llamar "no productivas" se ha centrado en la integración de librerías externas a .NET al proyecto: Emgu CV y J2KSession. Se trata de los típicos problemas de rutas de ficheros o variables de entorno, que aunque predecibles, su depuración suele llevar demasiado tiempo.

Hubo un caso de complejidad real y fue la sincronización por parte de los hilos que gestionaban cada nueva imagen que llegaba a la aplicación cliente desde el servidor. Estos hilos no podían acceder a la interfaz de usuario y de esa forma establecer las nuevas fuentes de imagen en el control específico para poder ser mostrada esta nueva imagen.

Esto es así porque en .NET únicamente el hilo principal, el que ha creado la interfaz de usuario (el primero que se creó), tiene permisos para modificarla. Por tanto, para poder acceder desde estos otros hilos al control se probaron varias alternativas hasta que se llegó a la mejor solución. Gracias a la ayuda de Álvaro Luis Bustamante fui capaz de resolver la mayoría de los problemas que me impedían continuar.

5.7. Cámara PTZ

Las cámaras PTZ describen toda una categoría de cámaras de alta calidad. El acrónimo PTZ significa *pan-tilt-zoom* y se refiere a la capacidad de este tipo de cámaras de moverse en sentido horizontal (pan), en sentido vertical (tilt) y de modificar la distancia focal (zoom).

Las cámaras PTZ pueden moverse por tanto horizontalmente, verticalmente y acercarse o alejarse de un área o un objeto de forma manual o automática. Todos los comandos PTZ se envían a través del mismo cable de red que la transmisión de vídeo. Algunas de las funciones que se pueden incorporar a una cámara PTZ incluyen:

- Estabilización electrónica de imagen (EIS): La estabilización electrónica de la imagen (EIS) ayuda a reducir el efecto de la vibración en un vídeo.
- Máscara de privacidad: Permite bloquear o enmascarar determinadas áreas de la escena frente a visualización o grabación.
- Posiciones predefinidas: Muchas cámaras PTZ y domo PTZ permiten programar posiciones predefinidas, normalmente entre 20 y 100.
- Seguimiento automático: Es una función de vídeo inteligente que detecta automáticamente el movimiento de una persona o vehículo y lo sigue dentro de la zona de cobertura de la cámara.

5.7.1. Características de la cámara

La cámara PTZ que se usará para esta prueba de concepto es la Sony EVI-D100P, cuya fotografía se incluye a continuación:

Ilustración 99 Cámara PTZ del laboratorio GIIA

Sus características son las siguientes, según las especificaciones de Sony:

- **Funcionamiento silencioso**: Los motores de transmisión directa permiten la reducción de engranajes de transmisión, gracias a lo cual se reduce enormemente, en comparación con modelos convencionales, el ruido del movimiento de pan/tilt.
- **Zoom de 40 aumentos** (zoom óptico de 10x y zoom digital de 4x): Objetivo con zoom de enfoque automático, rápido y estable, que alcanza 40 aumentos y ofrece un funcionamiento "manos libres" con una calidad de imagen constante.
- **Objetivo de conversión incorporado para gran angular (65°):** Un nuevo objetivo gran angular permite capturar imágenes con un amplio campo de visión, convirtiéndola en una cámara ideal para salas pequeñas y aplicaciones de videoconferencia.
- **Cabezal con amplia capacidad y velocidad de pan/tilt:** La cámara EVI-D100/P se mueve de forma inmediata a posiciones determinadas gracias a uno de los dispositivos de pan/tilt más veloces disponibles.
- **Completamente controlable a distancia mediante un cable RS-232C (protocolo VISCA):** La configuración de la cámara y las funciones de pan/tilt/zoom (PTZ) se controlan a distancia vía PC
- **Unidad de control remoto por infra-rojos:** Unidad de control a distancia, fácil de utilizar, para las funciones de pan/tilt/zoom de la cámara.
- **Función automática de "sleep**": La cámara tiene la opción de apagado automático cuando no se utiliza durante un periodo específico de tiempo.
- **Seis posiciones de pre ajuste mediante batería de reserva:** La batería de reserva almacena los pre ajustes de las posiciones de pan/tilt/zoom, el modo de exposición automática y el modo de balance de blancos, incluso cuando la cámara está apagada.
- **Diversos efectos de imagen:** Inversión positivo/negativo, blanco y negro (imagen monocroma), entre otras.
- **Diseño de calidad superior:** Un diseño práctico que permite montar y desmontar la EVI-D100 fácilmente en caso de una posible reparación.

Ilustración 100 Sony EVI-D100P

La cámara es controlada mediante el programa servidor a través del cable usando el protocolo VISCA.

5.8. Manual de usuario

La aplicación produce el vídeo resultante de una conversación entre varios interlocutores situados en frente del sensor Kinect. Cada vez que uno hable, Kinect debe localizar el origen el audio y enviar un comando a la cámara PTZ para que enfoque a esa dirección.

Cuando la grabación finalice, el usuario puede firmar el vídeo digitalmente mediante un certificado digital si lo desea. En este apartado se van a describir las instrucciones necesarias para generar y firmar este fichero de vídeo, objetivos finales de esta prueba de concepto.

5.8.1. Instalación

Para instalar este programa en otro equipo distinto al equipo del laboratorio donde se ha instalado por defecto, es necesario tener instalado en la máquina destino el software y controladores del dispositivo incluidos en ***Microsoft SDK for Kinect v1.5***.

Además, el sistema operativo debe ser de la familia Windows, siendo recomendable **Vista o 7**.

Para instalar el software, simplemente copie la carpeta donde reside el software en el equipo del laboratorio al nuevo equipo.

5.8.2. Iniciar la aplicación

Para iniciar la aplicación sólo es necesario ejecutar el programa **PrototypeKinectApp.exe** situado en la carpeta de instalación de la prueba de concepto.

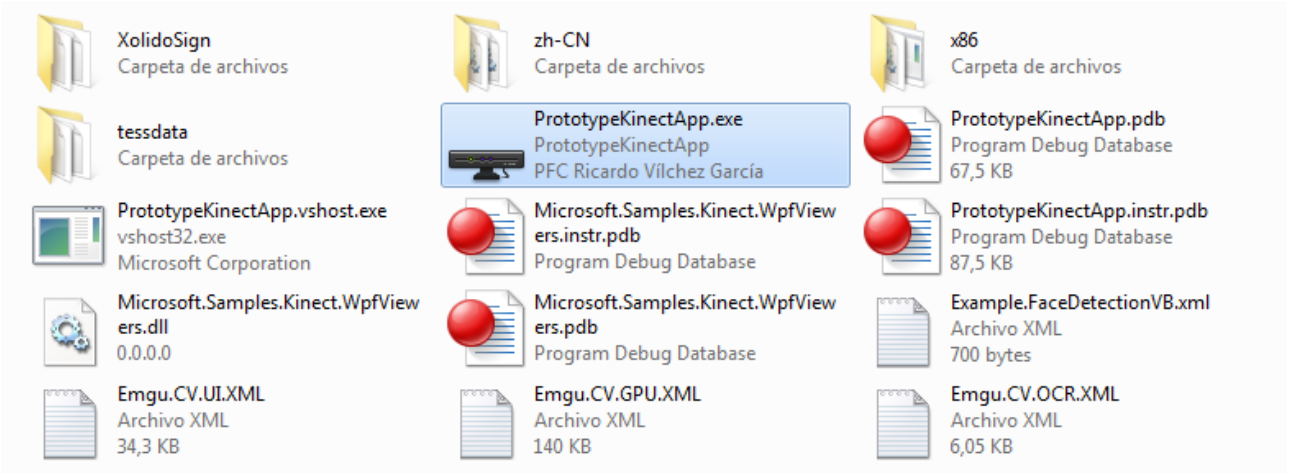


Ilustración 101 Ejecución del programa

Es necesario tener en cuenta que para que la aplicación funcione correctamente debe ejecutarse el programa desde la propia carpeta de instalación.

5.8.3. Interfaz de usuario principal

Antes de ejecutar el programa hay que asegurarse de que el sensor Kinect está conectado a la toma de corriente y al equipo donde se ejecute el programa mediante cable USB. El sensor Kinect debe mostrar su indicador LED parpadeando.

Si el sensor no está correctamente conectado al equipo o a una toma de corriente no se mostrará la imagen de la cámara RGB y se verá la siguiente notificación:

Ilustración 102 Notificación de Kinect no encontrado

Si el cable de USB está conectado pero falla la conexión a la toma de corriente, se mostrará el mensaje:

Ilustración 103 Notificación por falta de conexión a corriente

Cuando tanto el cable USB como la toma a corriente estén bien conectados, el siguiente mensaje cambiará y después desaparecerá, mostrando la imagen de la cámara RGB.

Ilustración 104 Notificación de inicialización de Kinect

Ilustración 105 Notificación de configuración correcta de Kinect

Si en algún momento posterior a la inicialización del sensor éste se desconectara tanto su conexión USB al equipo como a la toma de corriente, la ejecución del programa se detendría y se esperaría a que la conexión se hubiera restaurado.

Si la conexión del sensor es la adecuada, al ejecutar el programa se muestra la interfaz de usuario principal:

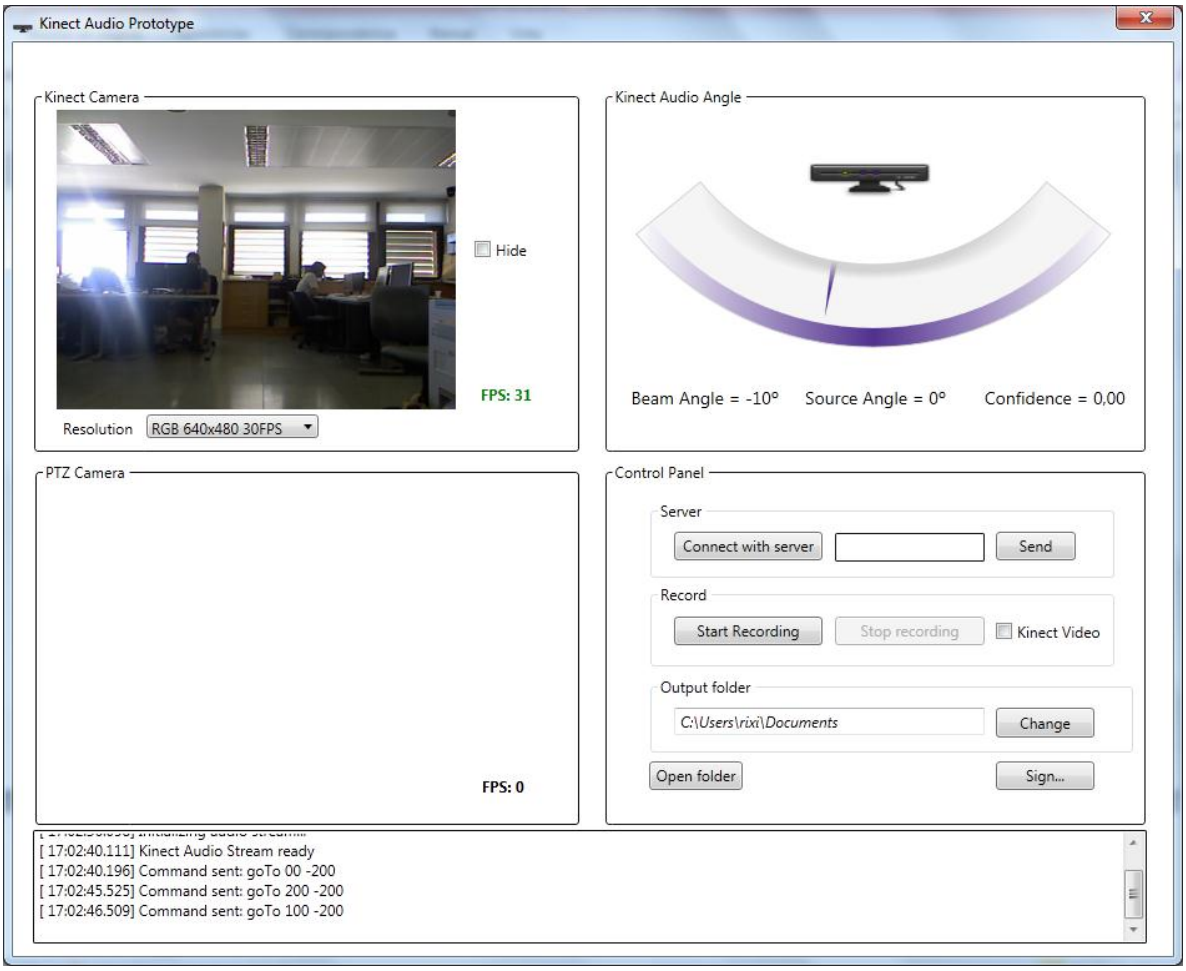


Ilustración 106 Interfaz principal de la aplicación

Los apartados en los que se divide son, según su posición:

- Superior izquierda: **Kinect Camera**. Muestra la captura de imagen recibida por la cámara RGB del sensor.
- Superior derecha: **Kinect Audio Angle**. Muestra la localización de fuente sonora calculada por el sensor, el grado de confianza de la precisión de ese valor y la sección o *beam* que incluye al ángulo (secciones de 10 grados).
- Inferior izquierda: **PTZ Camera.** Muestra la captura de imagen recibida del servidor y generada por la cámara PTZ. Sólo se muestra si se ha establecido una conexión con el servidor previamente.
- Inferior derecha: **Contol Panel**. Muestra los controladores de la aplicación y algunos parámetros de configuración.

Se trata de una aplicación estándar para Windows con el menú superior de control de ventana en el que aparece el título de la aplicación y el botón para cerrar la ventana con forma de "X".

Nada más arrancar la aplicación se inicializa el controlador del sensor Kinect y se observa la imagen de la cámara RGB en la sección Kinect Camera. La recepción de flujo de datos de audio, sin embargo, no comienza inmediatamente y **requiere de cuatro segundos para iniciarse**.

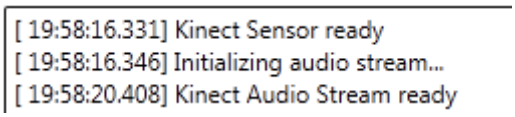


Ilustración 107 Inicialización del flujo de audio de Kinect

Una vez pasado este tiempo, tal y como muestra la interfaz, se empieza a recibir el flujo de datos de audio del sensor y la sección Kinect Audio Angle se activa, mostrando constantemente los cambios en la localización de fuente sonora.

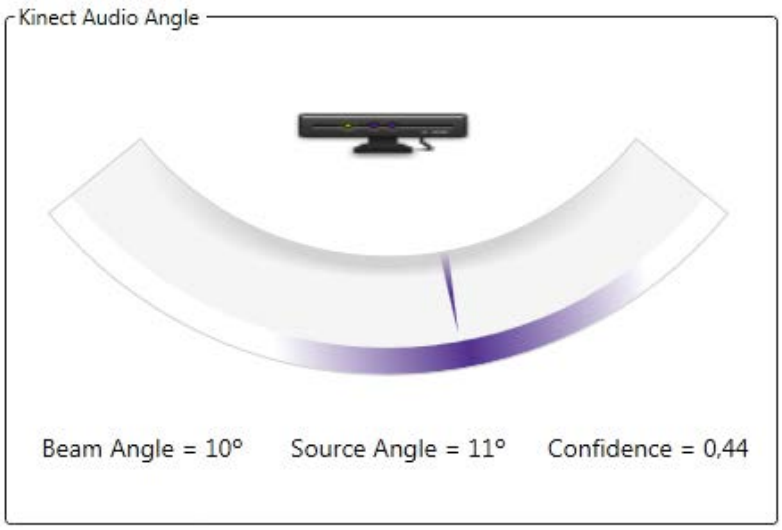


Ilustración 108 Sección Kinect Audio Angle en funcionamiento

5.8.4. Conectar con el servidor

Para establecer la conexión con el programa servidor y de esta forma empezar a recibir las imágenes capturadas por la cámara PTZ simplemente hay que presionar el botón “Connect with server”.

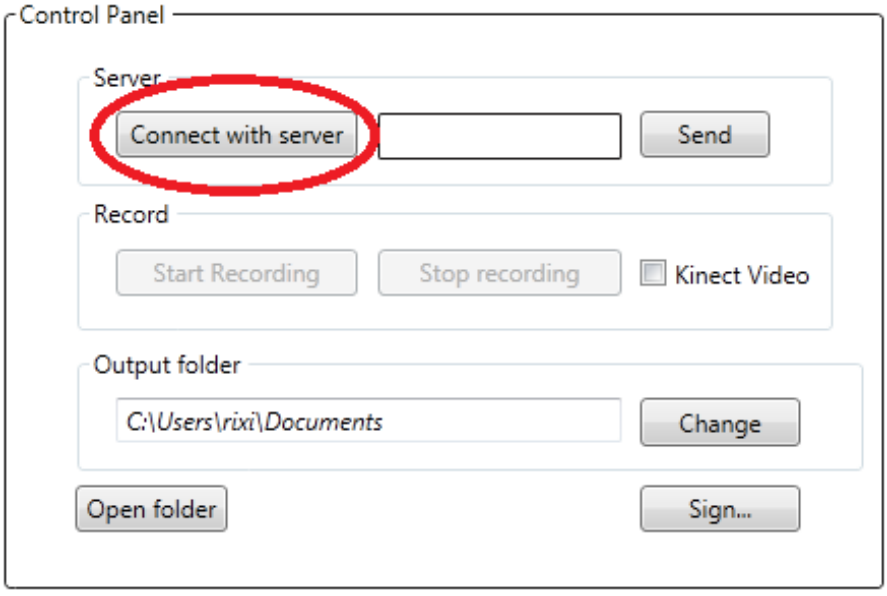


Ilustración 109 Iniciar la conexión con el servidor

En el registro (log) del programa vemos cómo se van produciendo las distintas fases de la conexión:

1. Inicio de la conexión RTP (Real Time Protocol)
2. Conexión a la dirección *multicast* de emisión streaming.
3. Conexión realizada

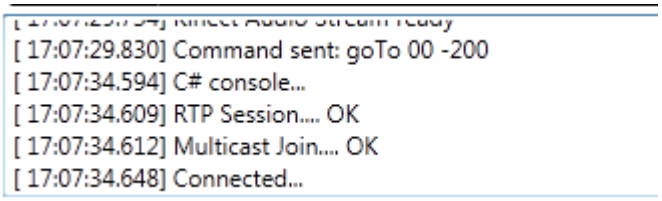


Ilustración 110 Información de conexión al servidor

5.8.5. Configuraciones previas

Antes de comenzar el proceso de grabación, si se desea, es posible modificar la carpeta de salida de los ficheros de la aplicación: El fichero de grabación de audio (denominado *audio_FechaDeGrabación.wav*), el de vídeo sin audio (*video_FechaDeGrabación.avi*) y el resultado final, el vídeo unido al audio (*recording_FechaDeGrabación.avi*).

Para cambiar la carpeta de salida, se presiona el botón “Change”.

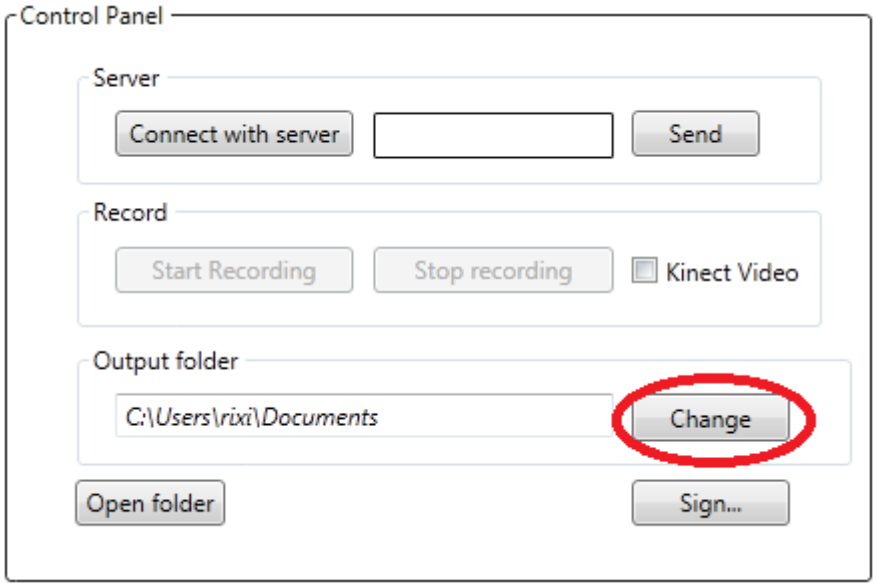


Ilustración 111 Cambio de carpeta de salida

A continuación se abre una ventana de explorador de Windows. Simplemente navegue por el sistema de ficheros del sistema y seleccione una carpeta.

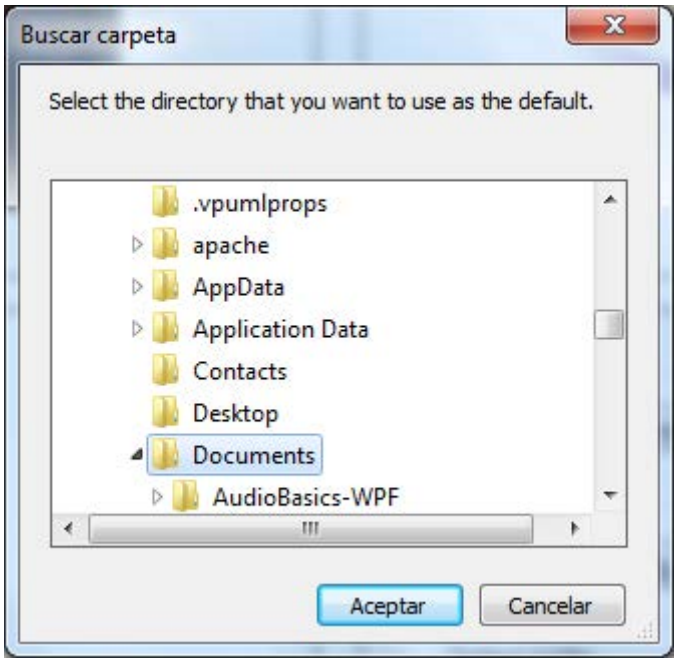


Ilustración 112 Selección de carpeta de salida

También es posible elegir el sensor Kinect como fuente de captura de imágenes para la generación del vídeo. Para realizar esto, simplemente hay que seleccionar la opción "Kinect Video".

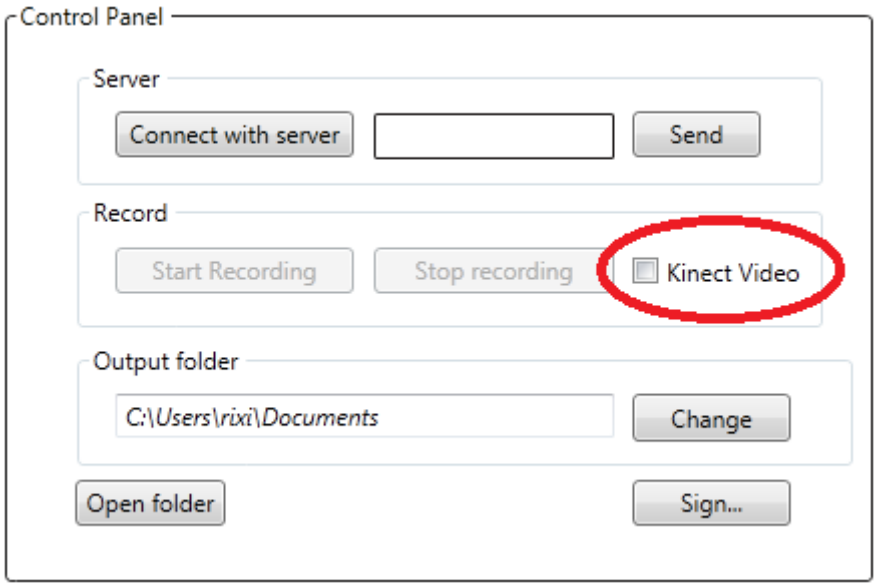


Ilustración 113 Opción Kinect Video

5.8.6. Iniciar la grabación

Para iniciar el proceso de grabación debe haberse establecido previamente la conexión con el servidor, a no ser que se haya seleccionado como fuente de video el sensor Kinect. De no ser así no se podrá almacenar ninguna imagen en el vídeo.

Para iniciar la grabación seleccione la opción “Start Recording”.

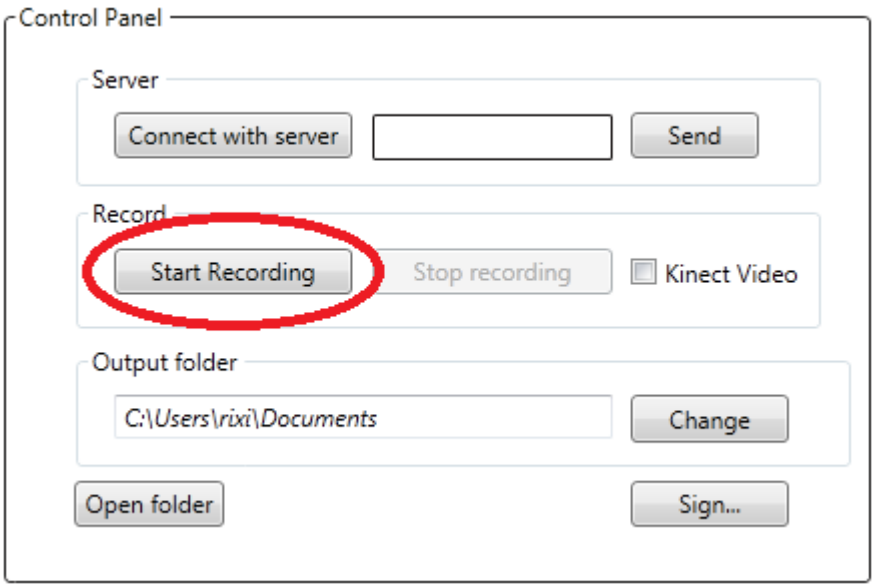


Ilustración 114 Inicio de grabación

5.8.7. Detener la grabación

Cuando el usuario desee finalizar la grabación, simplemente debe pulsar sobre el botón “Stop recording”:

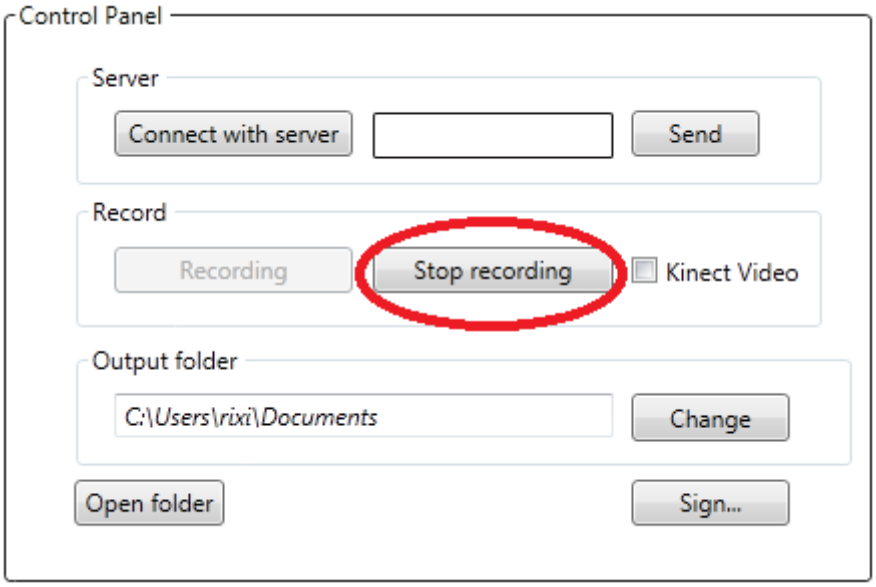


Ilustración 115 Detener el proceso de grabación

A continuación el programa finalizará la grabación de audio y vídeo, juntará ambos ficheros generados en un vídeo final. Se mostrará brevemente una consola de comandos, ya que el proceso de unión de ambos ficheros es realizado por el programa FFmpeg, que se ejecuta por consola.

Cuando finalice este proceso, se mostrará por pantalla un mensaje de notificación con el nombre del fichero final:

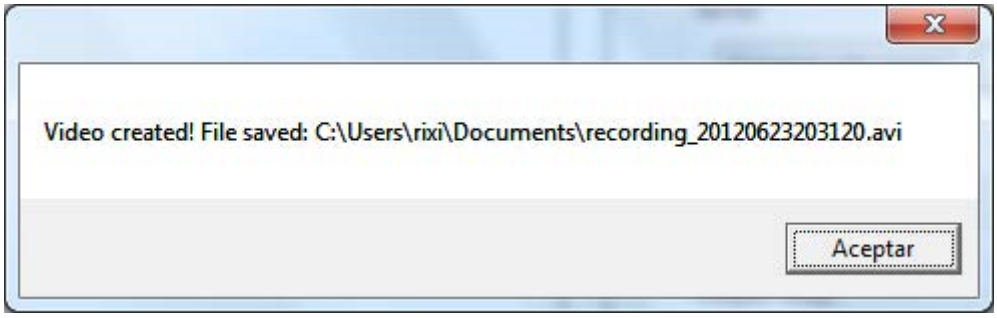


Ilustración 116 Mensaje de notificación. Proceso finalizado

El nombre del fichero resultante de la video acta se compone de "recording" y va acompañado de la fecha de creación. Esta fecha es la concatenación del año, mes, día, hora, minuto y segundo. La video acta del ejemplo de la ilustración 117 fue creada el 23 de Junio de 2012 a las 20:31:20.

5.8.8. Firmar digitalmente el vídeo

Si no se dispone de un certificado digital expedido por una Autoridad de Certificación no se podrá realizar este proceso. En este apartado se va a utilizar mi certificado personal expedido por la Fábrica Nacional de Moneda y Timbre.

Para ejecutar el programa externo XolidoSign hay que seguir los siguientes pasos:

1. Presionar sobre el botón "Sign…":

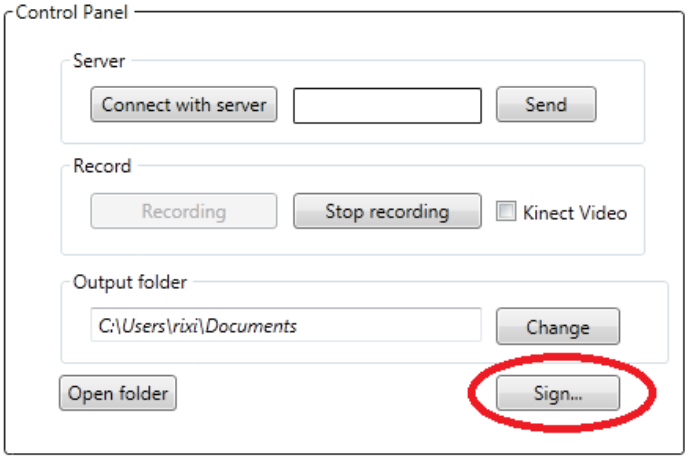


Ilustración 117 Llamada a XolidoSign

2. A continuación se abre la ventana principal de XolidoSign. Para firmar, seleccione a la izquierda la opción "Firmar".

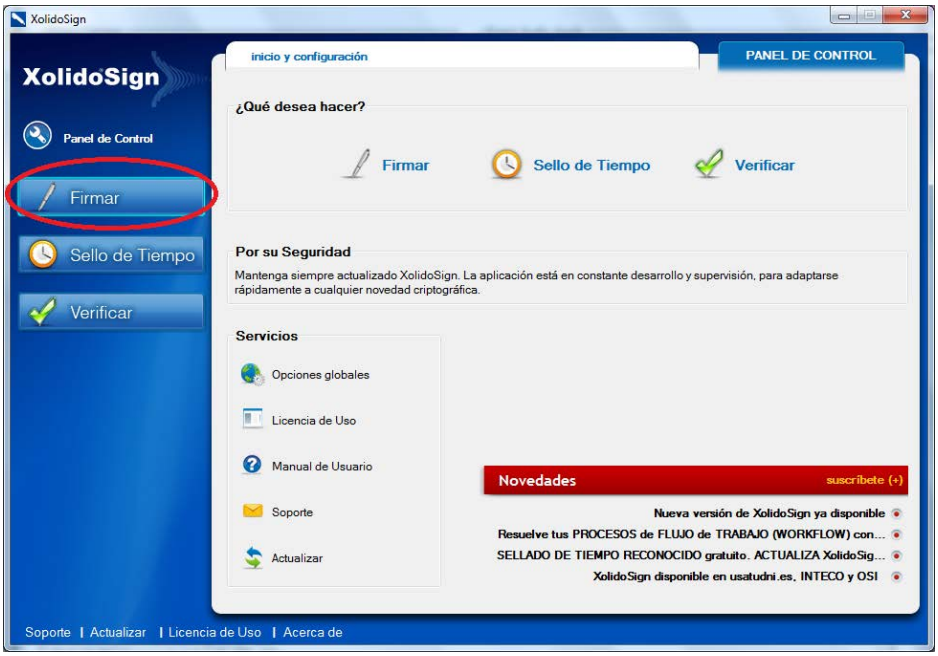


Ilustración 118 Ventana principal de XolidoSign

3. Después, seleccione el certificado digital con el que firmar.

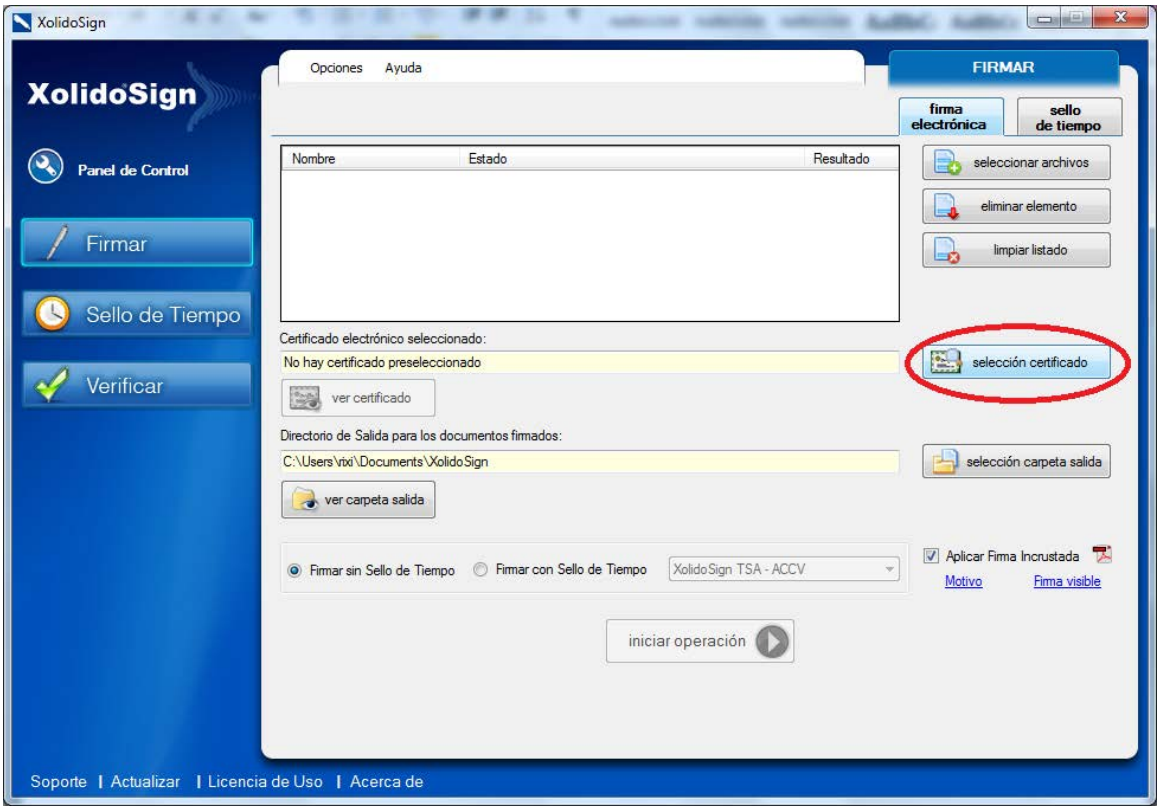


Ilustración 119 Selección de certificado digital (1)

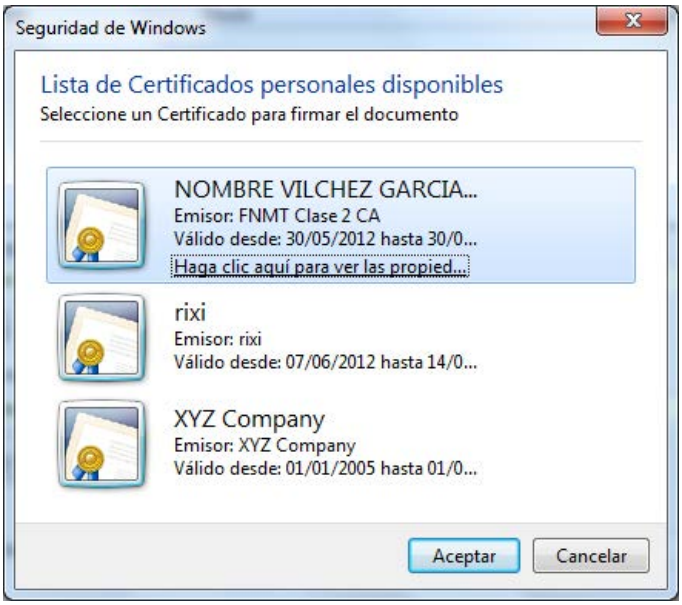


Ilustración 120 Selección de certificado digital (2)

4. Ahora la interfaz muestra su certificado seleccionado. Para añadir ficheros para firmar, seleccionar la opción "seleccionar archivos". Se mostrará una nueva ventana. Elija los ficheros que quiera firmar. En nuestro caso se seleccionará el fichero de vídeo generado con la aplicación.

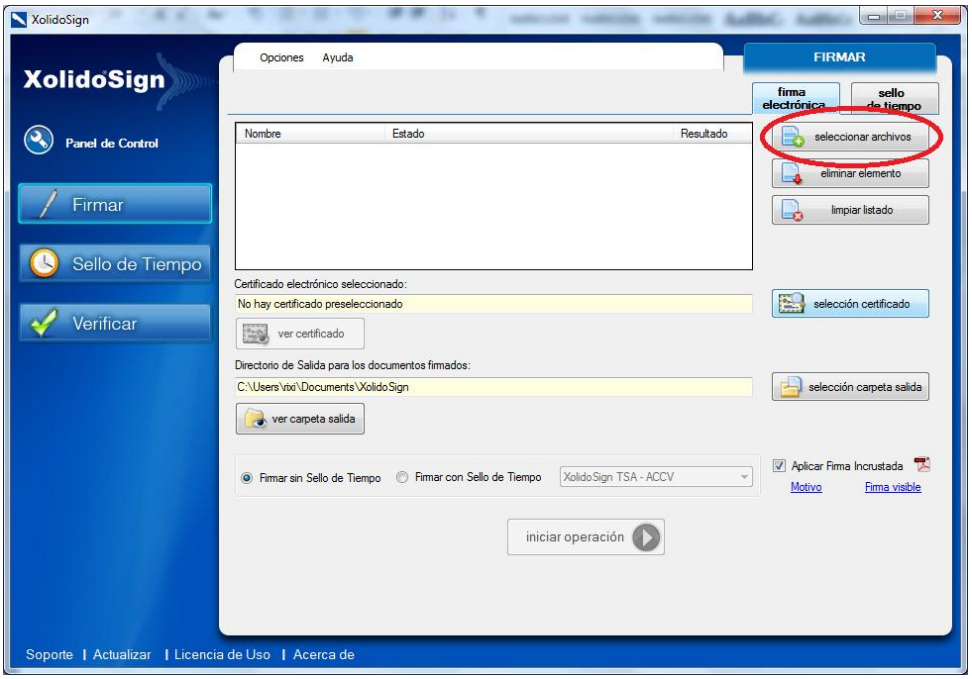


Ilustración 121 Seleccionar ficheros para firma

5. Para firmar el fichero, seleccionar el botón "iniciar operación".

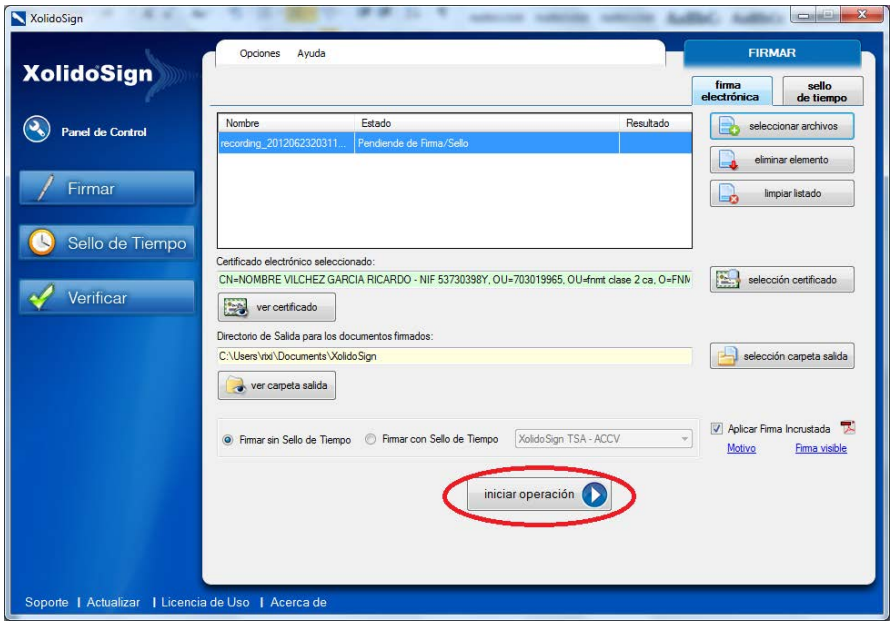


Ilustración 122 Ejecución del proceso de firma

6. Cuando finalice el proceso (dependiendo del tamaño del fichero) se mostrará un icono verde de finalización correcta en la columna “Resultado” de la lista de ficheros. Para ver el fichero de firma generado, seleccionar “ver carpeta salida”:

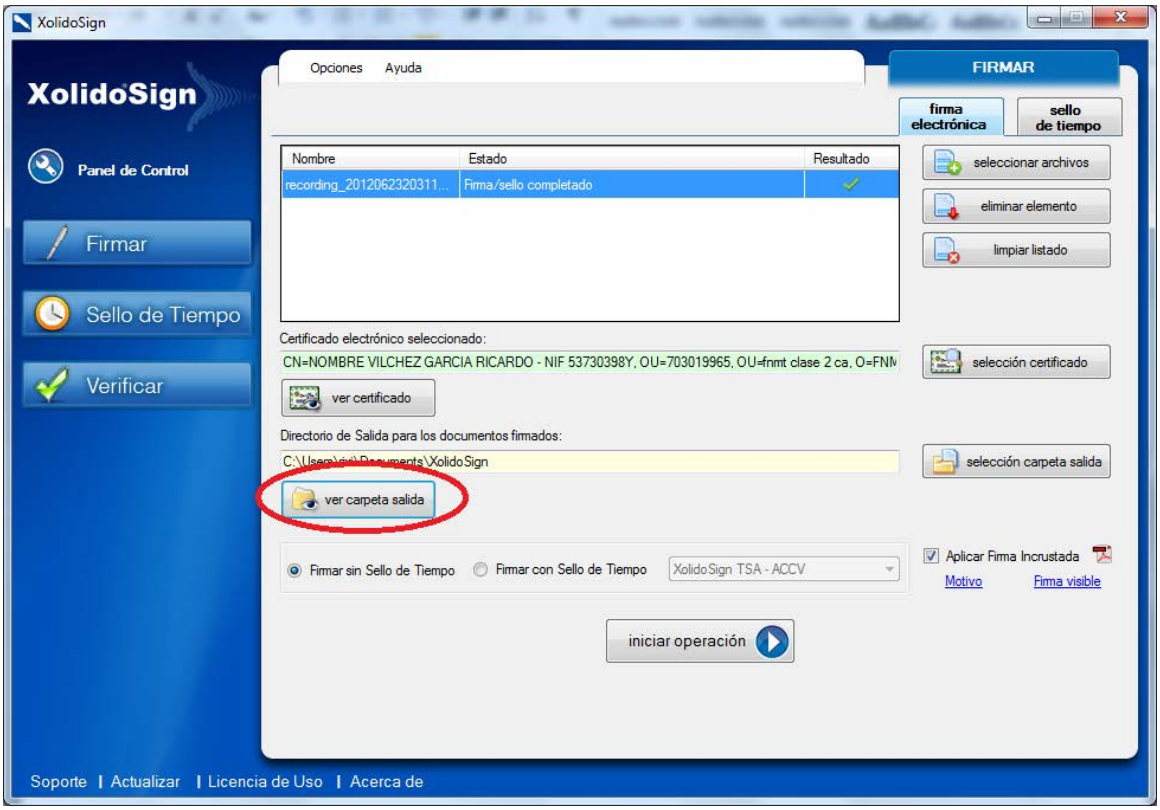


Ilustración 123 Abrir carpeta de salida

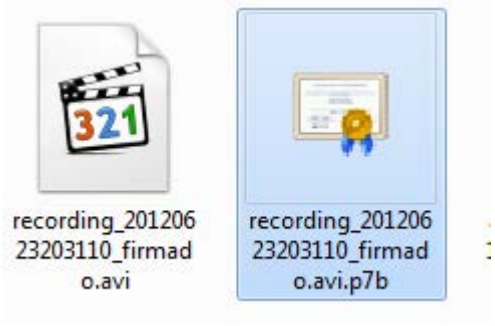


Ilustración 124 Fichero de vídeo y firma asociada

En la ilustración 125 vemos el fichero de firma generado por la aplicación así como una copia del fichero original al que se le ha añadido al nombre "_firmado".

5.8.9. Verificar la firma

La última funcionalidad que se muestra en este manual de usuario es la verificación de la firma digital obtenida en el apartado anterior (o cualquier otra firma asociada a un fichero concreto). Es importante disponer de tanto el fichero original sobre el que se realizó la firma como el fichero de firma en sí.

En la interfaz de usuario principal de XolidoSign, seleccionamos "Verificar":

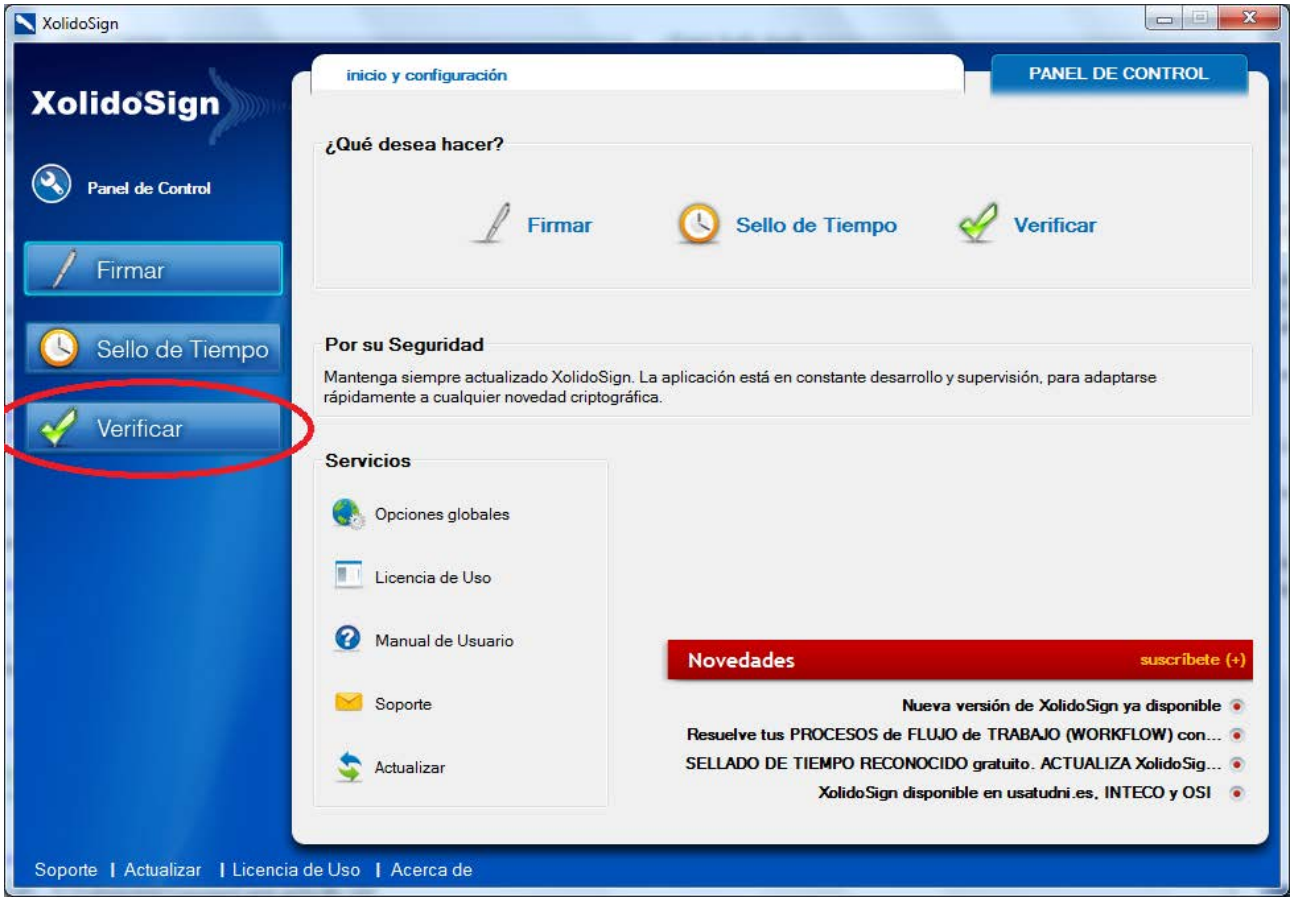


Ilustración 125 Selección de opción Verificar

Se abre la interfaz de verificación de firma electrónica. A la derecha, seleccionar “seleccionar archivos”. Se abrirá una ventana de navegación con carpeta raíz la carpeta de salida por defecto del programa XolidoSign. Si en el apartado anterior no se ha modificado esta carpeta, aquí se encuentran los ficheros generados.

XolidoSign tiene una funcionalidad que ellos denominan “verificación inteligente” que busca en la misma carpeta donde se encuentra un determinado fichero su firma asociada. Si la firma tiene otro nombre distinto al del fichero de vídeo, seleccionar en la interfaz la pestaña “verificación manual” y seleccionar el fichero de firma.

Por último, ejecutar la operación de verificación mediante el botón “Ejecutar operación”.

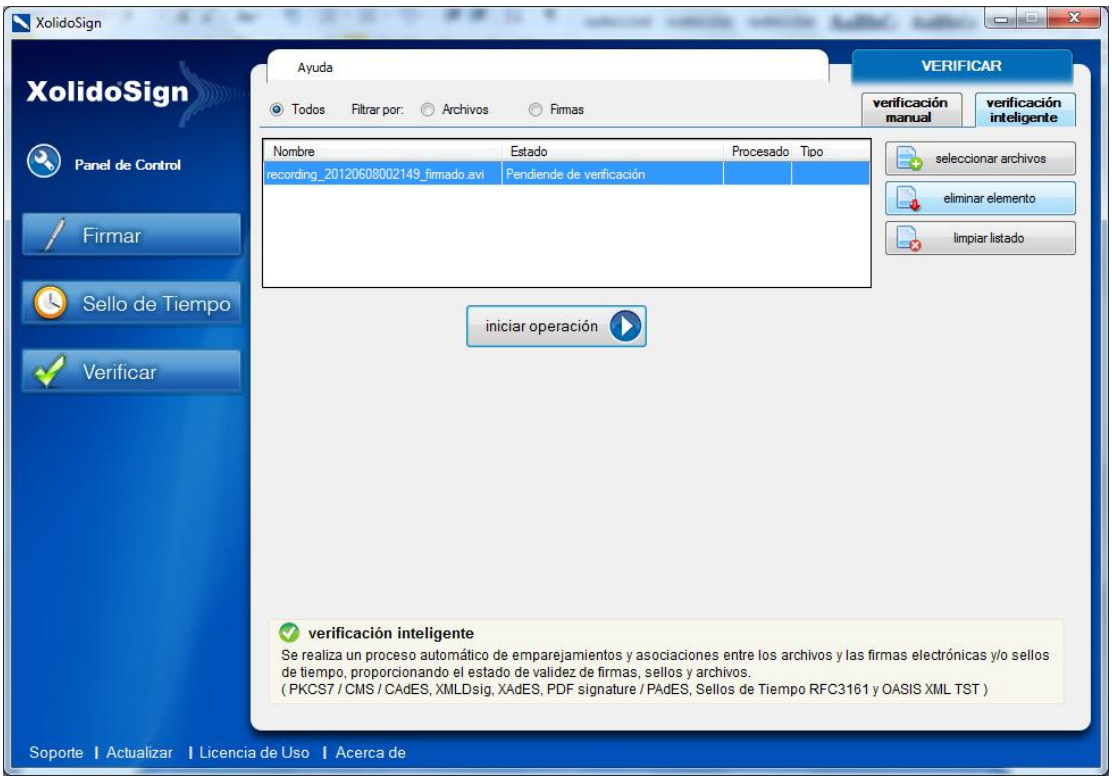


Ilustración 126 Archivos seleccionados

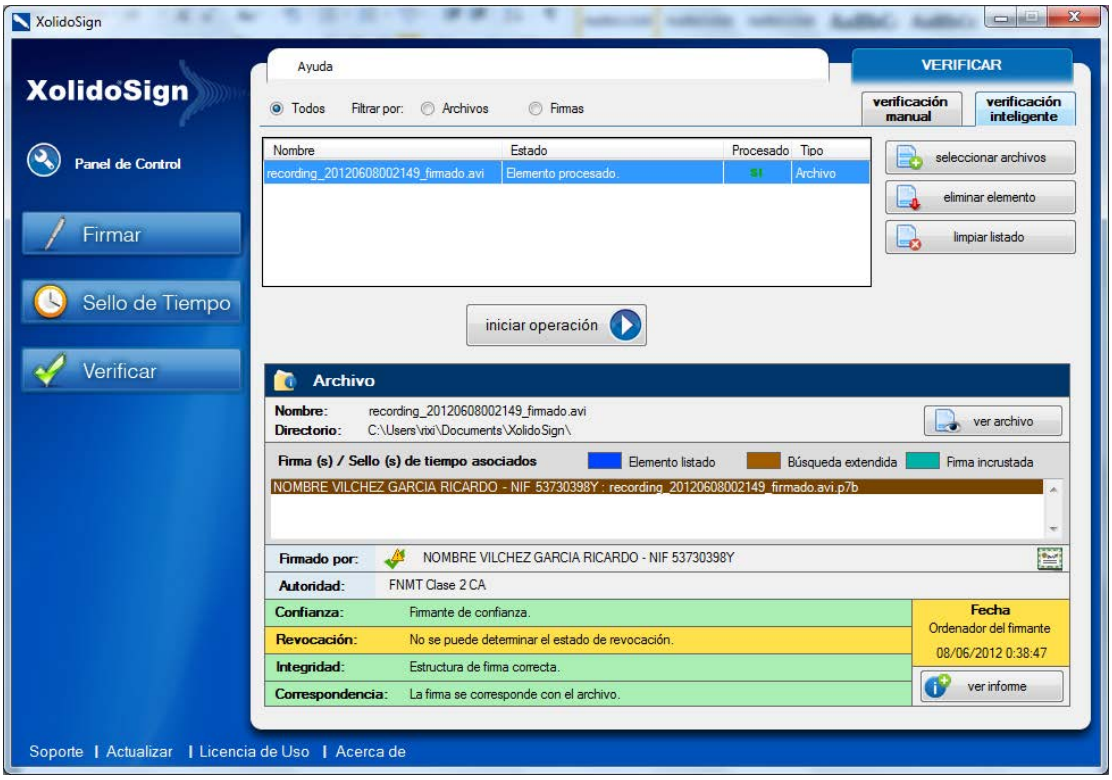


Ilustración 127 Resultado de la verificación de firma digital

El resultado de la verificación se muestra en la interfaz. Si la firma es correcta se muestra el texto sombreado en verde: El campo Integridad indica **"Estructura de firma correcta"** y el capo de Correspondencia indica **"La firma se corresponde con el archivo".** En conjunto indican, sin lugar a dudas, que el fichero no ha sido modificado desde que se firmó y la identidad de la persona firmante.

Se garantiza, por tanto, la integridad y la autenticidad del fichero.

6. Capítulo 6: Conclusiones

Finalmente, en este apartado se muestran conclusiones generales sobre el PFC, posibles líneas de futuro y la bibliografía usada.

6.1. Conclusiones generales

Bajo mi punto de vista, Kinect es un dispositivo fascinante porque ha sido capaz de traspasar las fronteras del mundo del ocio y se ha convertido en una potente plataforma de desarrollo (visión artificial, tele presencia, control gestual de dispositivos, etc.) a un coste más que razonable. Este dispositivo ha estado desde el principio enfocado oficialmente al mundo del ocio como dispositivo de control de Xbox 360 con el que se podía controlar un juego mediante nuestro movimiento o, incluso, con órdenes vocales.

Kinect, independientemente de que sea un producto de Microsoft, es un sistema que ha abierto una puerta hacia la innovación y la creatividad a muchas empresas y emprendedores; gente que ha visto que un dispositivo de 150 dólares era la pieza que necesitaban para desarrollar complejos sistemas de control remoto y reconocimiento de gestos o de comandos de voz.

En esta prueba de concepto hemos analizado el estado del arte de Kinect y hemos comprobado como estas labores de investigación se han centrado en explotar las funciones de imagen del sensor, dejando en un segundo lugar las de audio.

Al comienzo del presente documento se propusieron varios objetivos. Todos y cada uno de ellos se han cumplido con gran satisfacción por mi parte.

En los capítulos 2 y 3 se analizó el estado del arte del sensor Kinect y se describieron sus funcionalidades tal y como son accesibles desde el SDK que Microsoft pone al alcance de los desarrolladores. Se han estudiado las funciones de audio de cancelación de eco, supresión de ruido y localización de fuente sonora y cómo usando el SDK es posible obtener la dirección desde la que una persona está hablando.

En el capítulo 4 se realizó un análisis de la sensibilidad del sensor Kinect y se comprobó su estabilidad y precisión. Aunque muestra pequeñas desviaciones a los valores reales, la conclusión es que Kinect está perfectamente capacitado para desempeñar la tarea de localización de fuente acústica.

Según este resultado se usó esta funcionalidad para desarrollar una prueba de concepto con el objetivo de realizar un proceso de video acta.

Se considera por tanto que se han cumplido todos los objetivos propuestos para este trabajo.

6.2. Líneas de futuro

Como posibles mejoras de la prueba de concepto, se pueden ampliar los siguientes frentes:

- Uso del seguimiento de esqueleto en la identificación de fuente sonora.
- Uso de reconocimiento facial para identificación de personas en el escenario.
- Inclusión de más opciones: Formatos de vídeo, resoluciones, filtros, etc.
- Creación de un sistema distribuido de cámaras PTZ para generación de video acta.

A parte de la aplicación de video acta, ha quedado claro que Kinect ha permitido y continuará permitiendo a los desarrolladores idear nuevos proyectos que se beneficien de los elementos de que dispone. Espero también que mi humilde aportación haya servido para mostrar algo de luz sobre las funciones de audio del sensor.

6.3. Bibliografía

A lo largo del presente documento se han usado una gran cantidad de recursos obtenidos mayoritariamente de la Web y que me han ayudado a redactar y documentar muchos apartados.

A continuación se expone la lista de las referencias más representativas.

Libros:

- *Beginning Kinect Programming with the Microsoft Kinect SDK*, Jarret Webb y James Ashley. Editorial Apress. http://www.scribd.com/doc/94990688/Beginning-kinect-programming-with-the-microsoft-kinect

Fuentes electrónicas:

- Coding4Fun Kinect channel: http://channel9.msdn.com/coding4fun/kinect
- Kinect for Windows – Official Site: http://www.microsoft.com/en-us/kinectforwindows/
- Kinect resources: http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx
- Kinect for Windows documentation: http://msdn.microsoft.com/en-us/library/hh855347.aspx
- Kinect and education: http://www.kinecteducation.com/blog/2011/07/11/9-incredible-developments-for-kinect-in-education/
- Kinect for Windows SDK v1.5: http://blogs.msdn.com/b/kinectforwindows/archive/2012/03/26/what-s-ahead-a-sneak-peek.aspx
- Tecnología de Kinect: http://www.tierragamer.com/index.php/conoce-como-funciona-kinect/
- Kinect field of view: http://www.iheartrobotics.com/2010/12/limitations-of-kinect.html
- Kinect hacks: http://kinect.dashhacks.com/, http://www.kinecthacks.com/
- Comparación Microsoft C# y Oracle Java: http://www.25hoursaday.com/CsharpVsJava.html
- Firma digital con C# y .NET: http://tutorial.visualstudioteamsystem.com/details.aspx?item=134
- Manuales WPF: http://msdn.microsoft.com/en-us/library/ee649089.aspx
- Kinect Audio Beam: http://xenon.arcticus.com/kinect-z-buffer-noise-and-audio-beam-steering-precision
- Human-Computer Interaction: http://hci.uni-konstanz.de/blog/2011/03/15/navi/?lang=en
- Beamforming: http://www.xavieranguera.com/phdthesis/node39.html
- Gestos con Kinect: http://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/2011/HMM_Kinect.pdf
- Código administrado y no administrado: http://forums.codeguru.com/showthread.php?370170-.NET-Framework-General-What-is-the-difference-between-managed-and-unmanaged-code
- Open CV: http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html

- Emgu CV: http://www.emgu.com/wiki/index.php/Main_Page
- UDP sockets en C#: http://msdn.microsoft.com/es-es/library/1w48w47c(v=vs.80).aspx
- Multihilo en C#: http://msdn.microsoft.com/en-us/library/aa446528.aspx#multithreaded_netcf_apps_topic5
- Sinadura: http://www.sinadura.net/
- XolidoSign: http://www.xolido.com/lang/
- FFmpeg: http://ffmpeg.org/
- Cámara PTZ: http://www.sony.es/biz/product/ptzcams/evi-d100p/overview

7. Capítulo 7: Gestión del proyecto

En este apartado se describe el proceso que se ha seguido para desarrollar este trabajo. Como todo proyecto, se requiere una gestión adecuada. Se debe realizar una planificación donde se divida el trabajo a realizar en tareas y subtareas para que el desarrollo del proyecto se produzca de forma organizada.

Además se incluye una estimación del presupuesto necesario para desarrollar este proyecto.

7.1. Planificación

De cara a la planificación de un proyecto con estas características, el hecho de tener unos objetivos bien especificados desde el comienzo es una gran ventaja. Al no haber tenido que ir modificando objetivos o incluso encontrarlos por el camino, la planificación de este proyecto es bastante sencilla.

Ya que se trata en su mayoría de un proyecto de investigación, se asume un trabajo importante de documentación. Para este proyecto se trata de casi la mitad del tiempo dedicado, tal y como se mostrará a continuación. Las otras tareas realizadas incluyen la realización del análisis de sensibilidad, la prueba de concepto y la realización del presente documento.

A continuación se muestra un despliegue de las tareas realizadas.

Nombre	Fecha de inicio	Fecha de fin	ID	Duración
Documentación previa	13/02/12	9/03/12	0	20
Documentación Kinect sensor	13/02/12	24/02/12	7	10
Documentación Kinect SDK	27/02/12	9/03/12	8	10
Análisis de sensibilidad	12/03/12	23/03/12	2	10
Prueba 1	12/03/12	13/03/12	9	2
Prueba 2	14/03/12	16/03/12	10	3
Prueba 3	19/03/12	23/03/12	11	5
Desarrollo prueba de concepto	26/03/12	11/05/12	5	35
Documentación WPF	26/03/12	30/03/12	12	5
Documentación EmguCV	2/04/12	6/04/12	21	5
Desarrollo de interfaz de usuario	2/04/12	27/04/12	25	20
Imagen Kinect	4/04/12	9/04/12	13	4
Audio Kinect	2/04/12	6/04/12	14	5
Imagen cámara PTZ	2/04/12	6/04/12	18	5
Grabación video	9/04/12	19/04/12	19	9
Integración componentes	20/04/12	26/04/12	20	5
Pruebas	27/04/12	3/05/12	24	5
Instalación	4/05/12	11/05/12	22	6
Redacción de la memoria	12/03/12	15/06/12	6	70

Ilustración 128 Listado de tareas

Las fechas de inicio y de fin para cada tarea son estimaciones del tiempo real que se ha necesitado para realizar cada una. Estas estimaciones de plazos se consideran fieles al tiempo real empleado. Aparecen recogidas las fases de documentación previa, que al concluir permite iniciar la redacción de la memoria y el análisis de sensibilidad. Al finalizar este último, permite comenzar el desarrollo de la prueba de concepto.

La redacción de la memoria se considera entre las prioridades máximas, siendo la tarea más duradera en rango de tiempo. Nada más finalizar la documentación previa y hasta el fin del proyecto se realiza la redacción de la memoria, de forma que se va integrando en ella los distintos aspectos que se van tratando.

El resto de entradas de la ilustración 129 son subtareas en las que se dividen las principales tareas. Esta división es más perceptible en el diagrama de Gantt que se muestra a continuación.

En total se ha planificado un total de 90 días o 720 horas.

7.1.1. Diagrama de Gantt

Para facilitar la visualización no se incluyen los nombres de las tareas. La relación completa de tareas se detalla en la ilustración 129.

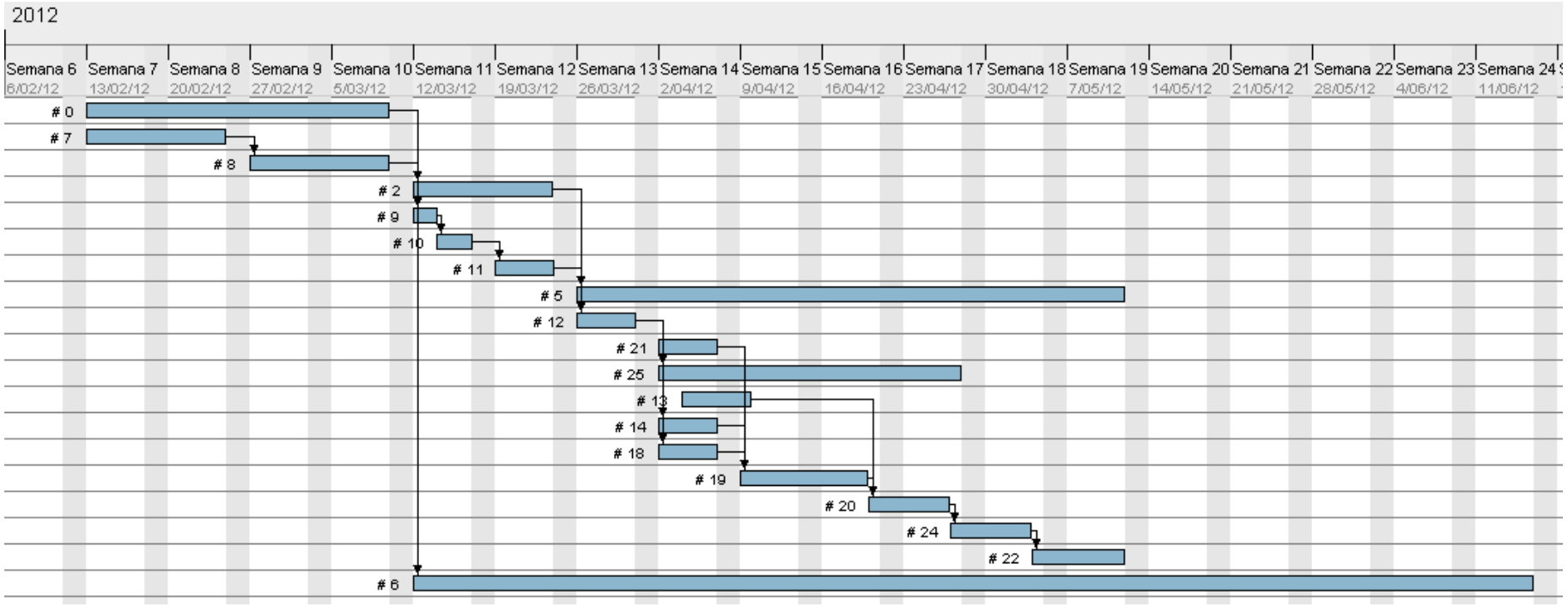


Ilustración 129 Diagrama de Gantt

Cada tarea y subtarea tiene asignada a su izquierda su número identificador, el mismo que aparece en la columna “ID” de la ilustración 129.

7.2. Presupuesto del proyecto

En este apartado se establece una estimación del presupuesto necesario para realizar el presente proyecto. El sistema a desarrollar es el descrito en la prueba de concepto: una aplicación que sea capaz de realizar, usando tanto el sensor Kinect como una cámara PTZ, una grabación de video acta para un escenario determinado.

Se realizará una evaluación de los recursos humanos y materiales (tanto hardware como software) necesarios para la realización del proyecto. El coste de los recursos humanos se realizará basándonos en la planificación expuesta previamente, y en la estimación de los sueldos que recibirán cada uno de los roles necesarios en la gestión del proyecto software.

7.2.1. Recursos Hardware

Se considera que el cliente de este proyecto dispone del escenario preparado para la integración de este nuevo sistema en sus instalaciones. Por tanto, se da por hecho que ya se dispone de:

- La cámara PTZ
- El servidor de *streaming*
- Red local Ethernet
- Equipo sobre el que se instala el sistema

El equipo en el que debe instalarse la aplicación cliente debe disponer de:

- Procesador Intel Pentium IV 3.4 GHz
- 2 GB de memoria RAM
- Disco duro de 120 GB
- Teclado y ratón
- Tarjeta de video
- Entrada para conector USB 2.0

Estos elementos deben tener al menos las mismas características que los elementos sobre los que se desarrolló la prueba de concepto. Se deja por tanto a cargo del cliente de proveerse del presupuesto de estos elementos.

Aparte de este hardware a cargo del cliente, únicamente es necesario incluir como requisito el propio **sensor Kinect para Windows**.

7.2.2. Recursos Software

Los productos software que se presentan a continuación requieren licencias que han sido adquiridas por medio de recursos de la universidad (por tanto ya han sido amortizadas) o disponen de una versión docente y educativa sin coste. Las aplicaciones usadas han sido las siguientes:

- Sistema operativo Microsoft Windows 7 Professional Edition® de 64 bits. Sobre este sistema operativo se ejecutará tanto la aplicación final como los programas necesarios para desarrollarla.
- Suite ofimática Microsoft Office 2010®: Conjunto de aplicaciones ofimáticas utilizadas para la redacción de este trabajo y diversos procesamientos de datos relacionados.
- Entorno de desarrollo Microsoft Visual Studio 2010®: Herramienta básica para desarrollar aplicaciones sobre .NET usando como lenguaje de programación C#.

- Microsoft Kinect for Windows SDK v1.5: El SDK comercial de Microsoft que incluye las librerías y herramientas necesarias para desarrollar aplicaciones con Kinect.
- XolidoSign: Programa de cifrado digital para el proceso de firma electrónica con certificado digital.

7.2.3. Recursos humanos

En este apartado se definen los roles de los componentes del equipo que desarrolle el sistema.

- **Jefe del proyecto:** Máximo responsable de la realización el proyecto. Debe encargarse de la planificación inicial y la supervisión del ajuste a dicha planificación, así como la estimación del coste del proyecto y el mantenimiento del mismo, garantizando los márgenes de calidad establecidos.
- **Analista:** Responsable del enlace entre los desarrolladores (diseñadores) y el cliente. Debe ser el encargado de plasmar las necesidades del cliente en requisitos, y de la verificación mediante la creación de un plan de pruebas de que dichos requisitos son respetados.
- **Diseñador:** Responsable del establecimiento de la arquitectura del sistema, de manera que éste se adapte lo mejor posible a las necesidades planteadas por los requisitos que recibe del análisis. Lleva esta arquitectura hasta un diseño detallado que puede incluso incluir pseudocódigo de las funciones de los diversos módulos
- **Programador:** Responsable de la codificación final del diseño en el lenguaje de programación C#. Debe someterse a unas reglas de estilo definidas en las fases previas, y es responsable de la correcta documentación del código de acuerdo a ellas.

Existen además determinados roles secundarios, como gestores de calidad, de documentación o de pruebas, que se consideran englobados en los anteriores, y que en cualquier caso todos ellos recaerán sobre Ricardo Vílchez García. El coste estimado asociado a cada perfil es el siguiente:

	Jefe de proyecto	Analista	Diseñador	Programador
COSTE	60€/hora	50€/hora	40€/hora	30€/hora

7.2.4. Coste del proyecto

Se debe estimar la dedicación en horas que hará cada uno de los roles sobre el tiempo total de desarrollo del proyecto, que en la planificación se ha estimado en 90 días laborales. Hay que tener en cuenta para el cálculo de horas que cada día laborable consta de ocho horas.

Para el proyecto a desarrollar es muy importante un estricto control y supervisión, debido a la facilidad de desviarnos del tiempo planificado (y por lo tanto el coste asociado) por la necesidad de formación y documentación adicionales o la variada casuística. Se considera por tanto que la dedicación del jefe de proyecto debe ser el 75% del tiempo del proyecto.

El análisis es un factor fundamental, para poder discernir los requisitos fundamentales, y desde ellos elaborar las mejores alternativas. Sin embargo este proceso se centra al comienzo del desarrollo únicamente y luego carece casi de importancia salvo modificaciones de requisitos. Se considera por tanto necesaria la dedicación del 30% del tiempo por parte del analista.

Los dos roles restantes presentan una menor importancia en el desarrollo del proyecto. La arquitectura del sistema es simple, y la interfaz de usuario de la aplicación no presenta demasiada complicación. Por ello se considera necesaria

una dedicación por parte del diseñador del 40% del tiempo total del proyecto. Por último, la fase estricta de programación es relativamente larga, por lo que se requiere una dedicación del 50% del tiempo total.

Como último añadido al coste del proyecto se considera necesaria una provisión para consumibles y gastos inesperados. Recordemos que los costes de software se pueden usar al usar licencias docentes, por lo que los costes del sistema se reducen a recursos humanos y recursos hardware (el sensor Kinect).

Los costes de hardware son:

Elemento	Precio
Sensor Kinect	200 €

Los costes de recursos humanos son:

Elemento	Porcentaje dedicación	Horas dedicación	Coste por hora	Coste
Jefe de proyecto	0.75	540	60 €/h	32400 €
Analista	0.3	216	50 €/h	10800 €
Diseñador	0.4	288	40 €/h	11520 €
Programador	0.5	360	30 €/h	10800 €

Añadiendo una provisión de 1000 € para gastos inesperados y materiales consumibles, tenemos juntos a los valores anteriores el valor final del coste presupuestado.

COSTE FINAL PRESUPUESTADO:
66.520 €